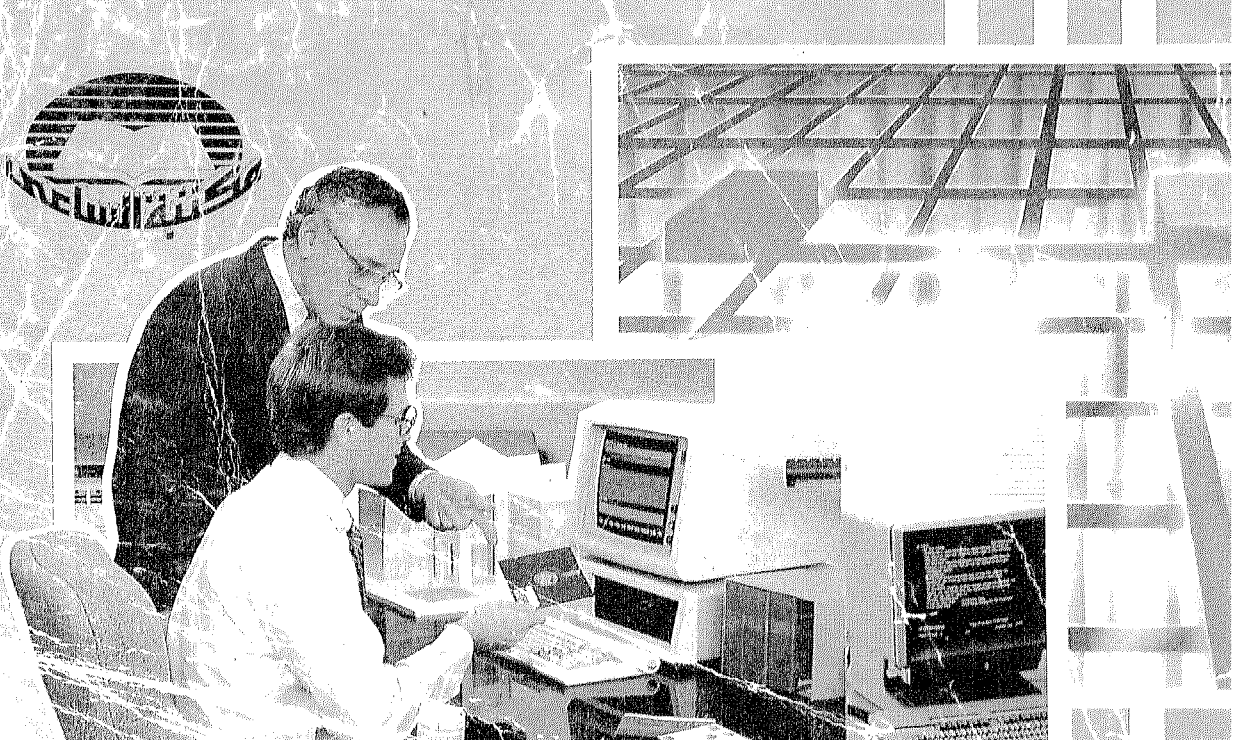
مهندس أسامة الحسيني
مصمم نظم بشركة تكنولوجي
انترناشيونال
الولايات المتحدة الأمريكية

تحدث إلى الكومبيوتر بلغة باسكال PASCAL

مكتبة الساعي



مهندس أسامة الحسيني
مصمم نظم بشركة تكنولوجى
انترناشيونال
الولايات المتحدة الأمريكية

تحدث إلى الكومبيوتر
بلغة
باسكال PASCAL

مكتبة الساعي
الرياض تليفون ٤٢١٥٦٣٦-٤٢١١٤٣٤
ص. ب. ٥٠٦٤٩ - الرياض ١١٥٣٣

بسم الله الرحمن الرحيم

بسم الله الرحمن الرحيم

كلمة المؤلف

باحثٌ عن الدهشة فى بلاد العم سام !

حملت عصاى وشددت الرحال إلى بلاد العم سام فكان أول ما بحثت عنه « أسرة الكومبيوتر » التى كنت أنتمى إليها فى مصر وفى مدينتى الاسكندرية لكن حقيقة هائلة أذهلتنى فى النهاية وهى أن الناس لا يحبون الكومبيوتر بنفس الكيفية التى نحبّه بها فى مصر . نعم هم يستخدمون الكومبيوتر ويعتمدون عليه فى العمل بصفة مطلقة . لكننى لم ألتق بشاب يحدثنى عن الكومبيوتر أو ناشىء يعشق البرمجة أو طفل يتعجل الخطى نحو التقدم فى التعامل مع الكومبيوتر ..

كانت أسرة الكومبيوتر فى مصر تضم الشباب والنشء والأطفال وأساتذة فى الجامعات يربطهم جميعاً حب الكومبيوتر والاهتمام بكل ما يخصّه من أخبار . حتى كان الحديث عن الكومبيوتر هو حديث المقهى والنادى وحديث الطريق العابر . وكم من أطفال فى مصر برعوا فى برمجة الكومبيوتر بدرجة تفوق أعمارهم ومستوى دراستهم .

أما الطفل الأمريكى فبرغم أنه يدرس مادة الكومبيوتر كمادة أساسية فى المدرسة لكنه ينظر إلى الكومبيوتر كما ينظر إلى شجرة فى الطريق . لا يدهشه الروبوط أو الكومبيوتر الذى يأتى بالأعاجيب .

ولعل السبب فى ذلك يرجع إلى أن الرجل الأمريكى لا يرى فى الكومبيوتر إلاّ وسيلة لإدارة أعماله وتسهيل مصالحه . فالحضارة الأمريكية لم تنشأ إلاّ نتيجة الاحتياج والبحث عن الحلول . والرجل الأمريكى لا يهتم أن يجيد لغة

خلاف اللغة الانجليزية لأنه لا يحتاج إليها .

وعندما تجولت باحثاً عن المحترفين ، عثرت أخيراً على « دراويش » الكومبيوتر لكننى وجدت السواد الأعظم منهم من الأجانب الذين جاءوا من مشارق الأرض ومغاربها سعياً وراء البحث العلمى والتطوير ، أو لعل الدهشة هى التى جذبت الكثيرين منهم إلى أضواء الحضارة الأمريكية .
وهمس فى أذنى أحد الأصدقاء قائلاً :

« إن الأمريكيين أنفسهم لا ينكرون أن حضارتهم قامت على أكتاف هؤلاء الأجانب الباحثين عن المغامرة والدهشة ! » ولعل هذه الكلمات جعلت منطق الأشياء يستقيم فى ذهنى من جديد ، فقد اعتقدت دائماً أن الدهشة هى البداية الحقة للإبداع . وأن الطفل المصرى الذى يستولى عليه « هوس » الكومبيوتر ويجعله ينفق آخر مليم فى جيبه لكى يقتنى برنامجاً أو كتاباً عن الكومبيوتر ، هذا الطفل بعينه سوف يكبر ذات يوم ويأتى منه ما أتى من جهابذة الباحثين الذين تدين لهم البشرية بالفضل والعرفان .

أسامة الحسينى



الباب الأول

القواعد العامة للغة باسكال

مفتتح

لعلها فكرة صائبة أن نبدأ بالتعميم ثم ننتقل إلى التخصيص فنبدأ بعرض الصورة الكاملة لبرنامج باسكال بهدف التعرف على شكله العام وخصائصه ثم نتدرج بعد ذلك فى تقديم المكونات الأساسية التى نبنى منها البرنامج : الكلمات والعبارات والمؤثرات والتعبيرات !

ولعل هذه الطريقة قد تنجح أكثر فى جذب القارىء الجديد على لغات الكومبيوتر الذى يتعجل أن يرى البرنامج المتكامل ولو كان برنامجاً صغيراً ، ولعله أيضاً يتعجل الجلوس إلى جهاز الكومبيوتر لتجربة البرامج ومشاهدة النتائج .

أما القارىء المتمرس فى اللغات الأخرى فلا أعتقد أنه سيشعر بالملل معنا وهو يتابع البرامج التمهيدية فسرعان ما نصل إلى الثوابت والمتغيرات وقواعد اللغة بكل تفعراتها !

(١ ـــ ١) لغة باسكال :

تتميّز لغة باسكال ببنائها المتفرّد . وأولى ملامح هذا البناء أن البرنامج المكتوب بلغة باسكال سهل الكتابة ، سهل القراءة لكونه مبنيّاً بناء طبيعياً متسلسلاً . أما الخاصية الثانية فهى الطريقة المتميزة التى تبنى بها البيانات والتى تعطى المبرمج درجات حريّة فى استخدام هذه البيانات . وبذلك فإن برنامج باسكال يتميز بصفة عامة بالوضوح وهى الصفة التى يجب أن تتميز بها اللغات عالية المستوى (high level languages) لتسهيل كتابة البرامج وإصلاحها .

وبقدر ما هى قويّة فى أدائها للأغراض المختلفة بقدر ما هى اقتصادية من ناحية الزمن المستغرق لتنفيذ البرامج وكذلك من ناحية السعة التى يشغلها البرنامج .

وقد أثبتت لغة باسكال فعاليتها ـــ مع ظهور الميكروكومبيوتر ، ومع ذلك فهى تستخدم فى الأجهزة الكبيرة (Mainframe) والمتوسطة (Mini) والصغيرة (Micro) بفوارق بسيطة بين الطرازات .

وقد سميت لغة باسكال بإسم العالم الفرنسى بليزباسكال BLAISE PASCAL (١٦٢٣ ـــ ١٦٦٢) الذى ابتكر آلة حاسبة ميكانيكية فى وقت مبكر . أما اللغة نفسها فقد قدمها الدكتور نيكلوس ڤيرت Niklaus Wirth رئيس قسم علوم الكومبيوتر فى معهد التكنولوجيا الفيدرالية بزيورخ ـــ عام ١٩٧٠ .

(١ ـــ ٢) ترجمة اللغات عالية المستوى :

تتميز اللغات عالية المستوى بأنها تقدم التسهيلات لحل المشكلة المعينة موضوع البرنامج بصرف النظر عن تفصيلات لغة الماكينة الثنائية (binary code) ولذلك فهى تصلح للمستخدم العادى الذى لا يريد أن يشغل نفسه كثيراً بما فى داخل الكومبيوتر !

وبالطبع فإن أجهزة الكومبيوتر لا تفهم اللغات عالية المستوى وتحتاج دائماً لمترجم والترجمة عادة تكون بأحد وسيلتين :

(أ) الترجمة التجميعية COMPILATION

(ب) الترجمة الفورية INTERPRETATION

والترجمة الفورية تتم سطراً بسطر (أو أمراً بأمر) كما فى لغة بيسك (BASIC) وهذا يتم بواسطة برنامج آخر يسمى المترجم الفورى (interpreter) يراجع معك البرنامج كلما أدخلت منه سطراً للكومبيوتر ويصحّح لك الأخطاء .

أما الترجمة التجميعية فتتم جملة واحدة للبرنامج ككل ويقوم بها المترجم التجميعى (COMPILER) وبعد الترجمة تحصل على نسخة جديدة من البرنامج مكتوبة بلغة الماكينة . أما إذا كان البرنامج محتوياً على أخطاء فإنك تحصل على تقرير بهذه الأخطاء ، وفى هذه الحالة فإنه يلزم إعادة ترجمة البرنامج بعد إصلاح الأخطاء .

ويسمى البرنامج المكتوب باللغة عالية المستوى « البرنامج المصدر » (source program) ، وأما البرنامج المكتوب بلغة الماكينة والناتج من عملية الترجمة فليسمى : البرنامج الهدف (object program) .

(١ ــ ٣) ترجمة لغة باسكال :

من قرأ كتبنا عن لغة فورتران أو لغة كوبول سوف يلاحظ بلا شك أننا خصصنا جانباً كبيراً من المناقشة لعملية الترجمة . ولكنه فى الحقيقة أن اللغة كلما زاد انتشارها وتعددت طرازاتها كلما أصبح من غير المجدى دراسة مترجم بعينه . ففى هذا الكتاب نهتم أكثر بعرض فنون البرمجة بلغة باسكال بصرف النظر عن طراز اللغة المستخدم وبصرف النظر عن نوع الكومبيوتر الذى تعمل عليه . أما خطوات الترجمة فهى عادة تكون مشروحة فى كتاب اللغة الذى تشتريه مع مترجم لغة باسكال المحفوظ على القرص المغنطيسى أو الشريط

المغنطيسى . وبعض طرازات لغة باسكال تحتوى على كل من الترجمة التجميعية والترجمة الفورية .

كما أنه مع أجهزة الكومبيوتر الشخصى:IBM والأجهزة المتوافقة معه ، نجد أنواعاً متقدمة من برامج الترجمة توّفر على المستخدم الجهد المبذول فى ترجمة البرنامج وتعفيه من هذا كله بإعطاء أمر واحد فقط ، هو أمر التنفيذ ! ويقوم البرنامج المترجم تلقائياً بترجمة وتنفيذ البرنامج . وأحد أمثلة هذا المترجم هو المترجم السريع TURBO PASCAL .

ويمدّك المترجم السريع بمحرّر خاص (editor) لتكتب عليه سطور البرنامج وبه كل الوسائل التى تسهّل عملية تحرير البرنامج من كتابة وحذف وإدماج للكلمات أو حفظ واستدعاء للبرامج والملفات . وتتم عملية الترجمة بالضغط على زر واحد هو الحرف C (اختصار Compilation) فتحصل على تقرير كامل بتفصيلات عملية الترجمة ورسائل خاصة عن الأخطاء الموجودة بالبرنامج . كما يمكن إعطاء أمر التنفيذ مباشرة بالضغط على الحرف R (اختصار Run) فتتم عملية الترجمة والتنفيذ متتابعتين إذا كان البرنامج خالياً من الأخطاء وإلاّ فتحصل على تقرير الترجمة ورسائل الأخطاء .

والشكل (١ ــ ١) يوضح برنامجاً صغيراً بلغة باسكال مكتوب على صفحة المحرّر ونرى أعلى الصفحة اسم البرنامج وعدّاد السطور وعدّاد الأعمدة . كما يوضح شكل (١ ــ ٢) التقرير الناتج من عملية الترجمة يعقبه تنفيذ البرنامج .

```
Line 9   Col 1   Insert   Indent   A:S123MOD.PAS

program chararrays(input,output);
VAR ch: array[1..20] of char;
    n,i:integer;

BEGIN
lowvideo;

    readln(n);
    for i:=1 to n do
    read(ch[i]);

highvideo;

    writeln;
    for i:=1 to n do
    writeln(ch[i]);
END.
```

عداد السطور — عداد الأعمدة — اسم القرص المغناطيسي — اسم البرنامج

سطور البرنامج تتالى على صفحات محرر لغة باسكال

شكل (١ ــ ١)

برنامج باسكال على أحد صفحات المحرّر (editor)

```
>
Compiling
  18 lines

Code:        000C paragraphs (   192 bytes), 0D1C paragraphs free
Data:        0004 paragraphs (    64 bytes), 0FD8 paragraphs free
Stack/Heap:  89A4 paragraphs (563776 bytes)

Running
5
abcde
a
b
c
d
e
>
```

عملية الترجمة — تقرير المترجم — عملية التنفيذ — البيانات المدخلة — المعلومات الخارجة بعد المعالجة

شكل (١ ــ ٢)

تقرير المترجم وتنفيذ برنامج باسكال

(١ ــ ٤) نظرة شاملة إلى برنامج باسكال :

قبل أن نغرق فى التفصيلات ، دعنا نلقى نظرة شاملة على البرنامج المكتوب بلغة باسكال حتى نتعرف إلى ملامحه العامة ولنبدأ بهذا البرنامج الصغير شكل (١ ــ ٣) :

```
program demo(output);
begin
    writeln('HI THERE..');
    writeln('MY NAME IS HAZEM OSSAMA AL-HUSSEINY');
    writeln('IT''S BEEN NICE TO MEET YOU');
    writeln('KEEP IN TOUCH...')
end.
```

شكل (١ ــ ٣)

أياً كان هدف البرنامج فسوف نراه دائماً فى هذا الهيكل العام حيث يبدأ بالسطر الذى يحتوى على اسم البرنامج . ثم تتالى عمليات المعالجة المختلفة بين البداية (begin) والنهاية (.end) وهذا البرنامج يقوم بطبع أربعة سطور متتالية على الشاشة كما هو موضح فى الشكل (١ ــ ٤) :

```
HI THERE..
MY NAME IS HAZEM OSSAMA AL-HUSSEINY
IT'S BEEN NICE TO MEET YOU
KEEP IN TOUCH...
```

شكل (١ ــ ٤)

وأول ما يلفت النظر عندما نلقى نظرة أولى على البرنامج ، هو هذه العبارات التى تحتها خط . وبالطبع فإن هذه الخطوط لا تعتبر ضمن البرنامج ولا تكتب العبارات بهذه الصورة ، وإنما قد وضعناها لإبراز بعض **الكلمات المحجوزة**

التى يختص بها برنامج باسكال والتى لا يجوز استخدامها فى غير أماكنها بالبرنامج .

ولنستهل قراءة البرنامج بالسطر الأول لكى ندّون بعض الملاحظات :

program demo(output);

يبدأ البرنامج بالكلمة المحجوزة program يعقبها اسم البرنامج demo الذى نختاره على هوانا . أما الكلمة output التى جاءت بين القوسين فهى تخبر الكومبيوتر بأن يستعد لطباعة بعض العبارات عندما نعطى أمر التنفيذ . وقد نلتقى بكلمة input فى نفس هذا المكان مع برامج أخرى ، وهذه الكلمة تخبر الكومبيوتر بأن يستعد لاستقبال بعض البيانات من العالم الخارجى (من مستخدم البرنامج) !

فإذا كان البرنامج يستقبل **البيانات** ويطبع **المعلومات** معاً سوف نجد كلمة input, output بين القوسين . والعبارة program وما يتبعها عبارة اختيارية يمكن الاستغناء عنها فى الكثير من الطرازات . فإذا انتقلنا إلى السطر الثانى من البرنامج نجد كلمة (begin) بمعنى « ابدأ » . وهى دائماً تكتب قبل **عمليات المعالجة** المختلفة التى قد يتضمنها البرنامج مثل الطباعة وقراءة البيانات وإجراء العمليات الحسابية إلى آخره . وينتهى البرنامج بالكلمة (.end) مصحوبة بنقطة النهاية .

أما عمليات المعالجة نفسها فقد كانت كلها عمليات طباعة باستخدام **العبارة (writeln)** . وقد جاءت المعلومات المطلوب طبعها بين علامتى اقتباس مثل :

writeln('HI THERE.. ');

وعلامة الاقتباس المستخدمة فى لغة باسكال هى علامة الاقتباس المفردة ، وهى نفسها العلامة apostrophe المستخدمة فى اللغة العادية . ولذلك إذا

كانت الجملة (الرسالة) المطلوب طباعتها تحتوى على علامة اقتباس كما فى السطر الثالث :

IT'S BEEN NICE TO MEET YOU

فإن هذا يتطلب استخدام علامتى اقتباس متتاليتين كما فى البرنامج .

('IT''S BEEN NICE TO MEET YOU');

ولعله يبدو ملفتاً للنظر استخدام الأقواس علاوة على علامات الاقتباس مع العبارة writeln .

أليس هذا بالكثير ؟ ...

هناك سبب لذلك . وهو أن عبارة الطبع لها استخدامات أخرى ، فهى قد تستخدم لطبع **الرسائل (messages)** أو **الحرفيّات (strings)** كما جاء فى هذا المثال ، وقد تستخدم لطبع الأعداد ، كما تستخدم أيضاً لإجراء بعض العمليات الحسابية قبل طباعة النتائج . وهذا يقدمنا لمثال آخر من برامج باسكال شكل (١ ــ ٥) :

```
program room(output);

begin
   writeln('your room needs ', 3*2, ' sq.m. of vinyl');
   writeln('and ', 2*(3+2)*3, ' sq.m. of wallcovering.')
end.
```

شكل (١ ــ ٥)

يقوم هذا البرنامج بحساب مساحتى الأرضية والحوائط لحجرة ذات أبعاد معيّنة بهدف تغطيتها « بالڤينيل » وورق الحائط ! فإذا دخلنا إلى صلب البرنامج مباشرة فإننا نلتقى بالعبارة writeln الأولى والتى تحتوى بين قوسيها على ثلاثة

أجزاء مختلفة وقد استخدمت علامة الفاصلة (comma) للفصل بين الأجزاء .

الجزء الأول هو :

your room needs

نلاحظ وجود هذه العبارة بين علامتى اقتباس وهذا يعنى أن كل المطلوب من الكومبيوتر هو طبع ما بين العلامتين دون أدنى تصرف .

فلنر الجزء الثانى :

3*2

جاء هذا الجزء بلا علامات اقتباس . معنى هذا أنه على الكومبيوتر أن يجرى عملية معالجة معينة قبل أن يطبع النتيجة . والمعالجة المطلوبة هنا هى ضرب العددين 3 ، 2 ثم طبع الناتج .

أما الجزء الثالث فقد أتى بين علامتى اقتباس لذلك سيطبع كما هو .

sq.m. of vinyl

لذلك نتوقع أن تكون نتيجة تنفيذ هذا السطر هى طباعة السطر التالى من المعلومات .

your room needs 6 sq.m. of vinyl

يقول هذا السطر إن حجرتك تحتاج إلى ٦ أمتار مربعة من « الفينيل » ! ربما يضيف الكومبيوتر بعض المسافات الخالية (spaces) قبل الرقم 6 بحسب طراز اللغة ولكن هذه هى النتيجة المتوقعة بصفة عامة . وكما نرى أن النتيجة كلها جاءت على سطر واحد وهذا هو تأثير العبارة writeln حيث يدل الحرفان ln على وجوب الانتقال إلى سطر جديد بعد طبع المطلوب .

عند تنفيذ العبارة التالية (السطر الرابع) سيقوم الكومبيوتر بطباعة السطر التالى :

and 30 sq.m. of wallcovering

وفى هذا السطر طبع الكومبيوتر الجزئين بين علامتى الاقتباس بدون تصرف . أما العملية الحسابية فقد حسب نتيجتها فكانت 30 .

وتقول هذه الفقرة : إن الحجرة تحتاج أيضاً إلى ٣٠ متراً مربعاً من ورق الحائط .

● **الأخطاء النحوية Syntax errors :**

نلاحظ فى هذا البرنامج ضرورة استخدام الفواصل (commas) والفواصل المنقوطة (semicolons) فى الأماكن المحددة لها بالضبط ، وإلاّ فإن الكومبيوتر يرسل لك رسالة إنذار بوقوع خطأ ما فى قواعد اللغة (syntax error) وبالطبع يتوقف الكومبيوتر عن التنفيذ إذا صادف خطأ مثل هذا (عادة تكتشف مثل هذه الأخطاء أثناء الترجمة) .

أما طريقة تنظيم البرنامج فلا تخضع لقواعد مشددّة فيمكننا أن نترك ما نشاء من المسافات الخالية بين الكلمات والأرقام والرموز .

ولكن القاعدة هنا أن تكون هناك مسافة واحدة على الأقل بين الكلمات .

كما أن لنا الحرية فى تنظيم سطور البرنامج بالطريقة التى تسهل علينا قراءته ومراجعته .

● **أخطاء التنفيذ Execution errors :**

ربما يكون البرنامج سليماً من ناحية قواعد اللغة وخالياً تماماً من الأخطاء النحوية ، لكن الكومبيوتر مع ذلك يتوقف عن التنفيذ مرسلاً لك رسالة أخرى عن سبب عدم استطاعته تنفيذ البرنامج .

قد يكون السبب هو وجود عدد ما فى الحسابات الناتجة لا يستطيع التعامل معه لكونه كبير جداً أكبر من حدود الكومبيوتر . أو قد يكون السبب وجود عملية حسابية معضلة مثل القسمة على صفر وهى تخرج عن حدود أى كومبيوتر .

مثل هذه الأخطاء التى تحدث أثناء التنفيذ تسمى أخطاء التنفيذ (execution errors) .

● **الأخطاء المنطقية Logical errors :**

بعد أن يتأكد المبرمج من خلو البرنامج تماماً من الأخطاء النحوية وأخطاء التنفيذ المحتملة يعطى الأمر بتنفيذ البرنامج فيقوم الكومبيوتر بطبع النتيجة على الورق أو على الشاشة فى ثوان فيسعد المبرمج نفساً ! ولكن نظرة واحدة إلى النتيجة تصيبه بخيبة أمل إذ يكتشف أن الكومبيوتر قد أجرى حسابات خاطئة وطبع نتيجة غير معقولة .. غير منطقية .. فماذا حدث ؟ .

بالطبع الذنب ليس ذنب الكومبيوتر فهو يفعل ما يؤمر به طالما خاطبناه باللغة الصحيحة التى يفهمها وفى حدود إمكانات أجهزته . هذه النوعية من الأخطاء تسمى الأخطاء المنطقية ولا يستطيع الكومبيوتر التعرف عليها ولذلك لا يتوقف عندها .

فمثلاً لو أن المبرمج قد استبدل علامات الجمع (+) بعلامة الطرح (−) فى التعبير الحسابى الوارد فى السطر الرابع أى أنه كتب التعبير (سهواً) بالصورة الآتية :

2*(3−2)*3

فإن نتيجة هذه العملية الحسابية سوف تكون الرقم 6 بدلاً من النتيجة التى كان يقصدها المبرمج وهى العدد 30 . وهذا بالطبع ليس ذنب الكومبيوتر فهو لا يعلم قصد المبرمج إذا هو أخطأ التعبير .

● **جدوى البرنامج :**

لا شك أن القارىء الجديد على لغات الكومبيوتر سوف يجد هذا البرنامج تافهاً فى هدفه ولعله يتساءل عن جدوى هذا البرنامج الذى نبذل جهداً فى إعداده لكى يخرج علينا فى النهاية بنتيجة عملية حسابية سهلة كان من الممكن أن نحسبها شفهياً دون الحاجة إلى الكومبيوتر « من أصله » !

وهذا حق ... ولكن مهلاً !

فحتى هذا البرنامج الصغير يمكنه أن يؤدى عملاً نافعاً لو أنه كان يؤدى نفس العملية الحسابية لمختلف الحجرات ذات الأبعاء المختلفة ...

وهذا يمكن أن يتحقق إذا جعلنا البرنامج يستقبل بيانات جديدة فى كل مرة ويختزنها فى **متغيرات** مثل طول الحجرة وعرض الحجرة وارتفاعها .

فلا يمكن أن نتصور بحال أن ننشىء برنامجاً لحساب مرتب شخص واحد ، فالبرنامج عادة يكون برنامجاً عاماً يحتوى على متغيرات مثلاً الاسم وعدد ساعات العمل وعدد الساعات الإضافية إلى آخره .. ويقوم البرنامج باستقبال القيم العددية للبيانات الخاصة بكل موظف ويحسب له صافى الأجر على حدة .

وفى برنامجنا الصغير يمكننا أن نمثل الطول بالمتغير length والعرض بالمتغير width والارتفاع بالمتغير height فيصبح البرنامج كما فى شكل (١ ـــ ٦) :

```
program roomsize2(input,output);

var length, width, height : integer;

begin

   writeln('type in length, width and height');
   read(length, width, height);
   writeln('your room needs ', length*width,
          ' sq.m. of vinyl');
   writeln('and ', 2*(length + width)*height,
          ' sq.m. of wall covering')
end.
```

شكل (١ ــ ٦)

عند تشغيل مثل هذا البرنامج سوف يسألنا أن ندخل طول وارتفاع وعرض الحجرة المقصودة فيقوم على الفور بحساب المساحات اللازمة من ورق الحائط وغطاء الأرضية وذلك بفضل العبارة **read** التى ظهرت فى السطر الثانى .

وقد ظهر بهذا البرنامج سطر جديد قبل بداية البرنامج مباشرة بدأ بكلمة **VAR** ، وفائدة هذا السطر هو **الإعلان** عن المتغيرات المستخدمة فى البرنامج . وسوف نشرح هذا الموضوع تفصيلاً فى الأجزاء القادمة .

بقيت ملاحظة هامة يجب علينا أن نذكرها ، فلا شك أن بعض الأصدقاء الجدد على لغة باسكال يتساءلون « هل من الضرورى أن نكتب برنامج باسكال بالحروف الصغيرة ؟ » فى الحقيقة أن هذه النقطة قد تختلف فيها أجهزة الكومبيوتر وطرازات اللغة المستخدمة . فمع بعض الطرازات قد يستلزم الأمر أن تكتب بعض الكلمات المحجوزة باستخدام الحروف الكبيرة (Capital or uppercase) أما بقية البرنامج فيكتب بالحروف الصغيرة (Small or lower case) . ومع ذلك فالأجهزة الحديثة والطرازات الحديثة من لغة باسكال أصبحت تتغاضى عن هذا كله وتتسامح فى مثل هذه الشكليات ،

فالبرنامج التالى منفذ على أحد أجهزة الكومبيوتر المتوافقة مع IBM ومكتوب بالحروف الكبيرة شكل (١ ـــ ٧) وهو نفس البرنامج الذى عرضناه من قبل .

```
PROGRAM ROOM(OUTPUT);

BEGIN
   WRITELN('YOUR ROOM NEEDS ', 3*2, ' SQ.M. OF VINYL');
   WRITELN('AND ', 2*(3+2)*3, ' SQ.M. OF WALLCOVERING.')
END.
```

شكل (١ ـــ ٧)

وبالطبع يخرج عن المناقشة ما جاء بين علامتى الاقتباس مع عبارة الطباعة writeln فالحرفيات قد تكتب بأى صورة تراها .

أما القاعدة التى لا يتسامح فيها مترجم باسكال فهى ضرورة فصل **العبارات** عن بعضها البعض بفاصلة منقوطة (semicolon) . لأن كل عبارة تمثل أمراً مستقلاً ينفذ على حدة . ولعلك تلاحظ أن الفاصلة المنقوطة لم توضع بعد العبارة الأخيرة لأنها لا داعى لها .

- **القواعد العامة لبرنامج باسكال :**

لا توجد قواعد كثيرة تحدد شكل البرنامج المكتوب بلغة باسكال مثل الكتابة على نموذج معيّن أو فى مكان معين من السطر ، بل يكفى أن تبدأ العبارة من أول السطر . وحتى يكون بناء البرنامج واضحاً يفضّل أن تبدأ العبارات من نفس المسافة دائماً .

وتُفصل كلمات باسكال بمسافة خالية أو أكثر أو برمز خاص أو بنهاية السطر .

ويحتوى البرنامج بصفة عامة على الحروف الأبجدية والأرقام والعلامات الخاصة وأى منها يسمى لبنة (character) ، ومن اللبنات تتكون الكلمات

(words) ومن الكلمات تبنى العبارات (statements) التى تمثل وحدات بناء برنامج باسكال .

(١ ـــ ٥) أسماء البيانات (Identifiers) :

إن أسماء البيانات هى أسماء الأوعية التى توضع فيها البيانات سواء كانت هذه البيانات أرقاماً مثل « الأجر » أو حرفيّات مثل « الاسم » و « رقم التليفون » ... إلى آخره . ومن درس لغة بيسك من قبل فقد عرف أسماء البيانات تحت اسم المتغيّرات (variables) ، أما فى لغة كوبول فهى تسمى (data-names) . وها هى أسماء البيانات تهل علينا تحت اسم جديد هذه المرّة وهو (identifiers) مع فارق واحد وهو أن أسماء البيانات فى لغة باسكال تستخدم لتسمية المتغيرات علاوة على طراز جديد من البيانات وهو الثوابت المسمّاه كما سنعرف فى الفقرات التالية .

والقواعد العامة التى تخضع لها أسماء البيانات هى :

(أ) يتكون اسم البيان من الحروف الأبجدية (**من A إلى Z**) أو من الأرقام (**من 0 إلى 9**) أو من كليهما بشرط أن يبدأ الاسم دائماً بحرف مثل :

netpay , telephone , x1

أما بالنسبة لاستخدام الحروف الصغيرة أو الكبيرة مع أسماء البيانات فإن طرازات اللغة تختلف فى ذلك كما ذكرنا من قبل ولذلك يجب الاسترشاد فى هذه النقطة بدفتر اللغة الخاص بالكومبيوتر المعيّن .

(ب) لا يجب أن يتضمن الاسم مسافات خالية .

(جـ) بالنسبة لطول إسم البيان ، فدائماً هناك حد أقصى (ويكون ٣٠ لبنة مع أغلب الطرازات) .

ومع ذلك فإن مترجم لغة باسكال لا يلتفت إلاّ إلى الحروف الثمانية الأولى

من الاسم .

فإذا استخدمت اسمين مثل :

temperate temperature

فإنهما يصبحان اسماً واحداً عند ترجمة البرنامج .

وبصفة عامة تفضل الأسماء الطويلة التى تشرح معناها ذاتياً بدلاً من الحروف المفردة .

(١ ـــ ٦) الكلمات المحجوزة (Reserved Words) :

تتميز لغة باسكال بقلة الكلمات المحجوزة (المحظور استخدامها كأسماء للبيانات) ، وهذه الكلمات تتميز بأنها تحمل معنىً معيّناً يفهمه الكومبيوتر على نحو ما مثل :

PROGRAM , IF , OR , THEN .

والملحق (أ) فى نهاية الكتاب يتضمن قائمة بالكلمات المحجوزة فى لغة باسكال .

(١ ـــ ٧) الكلمات القياسية (Standard Words) :

هناك مجموعة أخرى من الكلمات الخاصة السابق تعريفها للمترجم ، ورغم أنه يمكن استخدامها كأسماء للبيانات لكن هذا يفسد معناها الأصلى الذى ابتكرت من أجله لذلك لا يُنصح باستخدامها . وسوف يلى الحديث عن فائدتها فى الفقرات القادمة . والملحق (ب) يضم هذه النوعية من الكلمات .

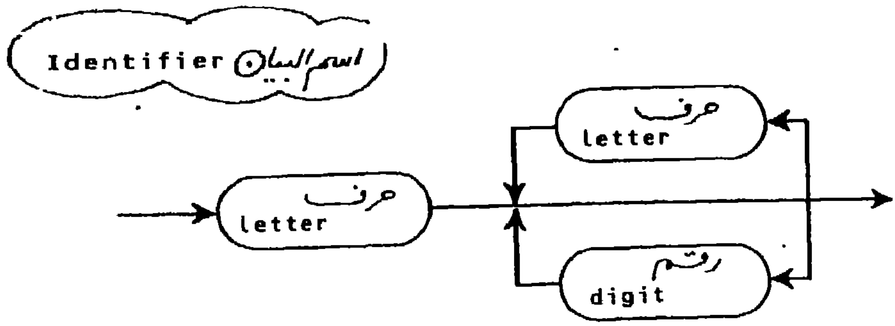
(١ ـــ ٨) خرائط قواعد اللغة (Syntax Diagrams) :

عادة تصاغ قواعد لغة باسكال بواسطة الخرائط وهى طريقة أكثر تعبيراً

وأسرع استقبالاً من القارىء .

وتستخدم فى الخرائط الحروف الكبيرة وعلامات الترقيم للتعبير عن أشياء قائمة بذاتها (مثل الكلمات المحجوزة وعلامات العمليات الحسابية) ، وتستخدم الحروف الصغيرة لتدل على المكونات الأخرى التى تتغير حسب الموقف مثل أسماء البيانات والثوابت .. وهذا مجرد عرف لكنه ليس ملزماً .

وشكل (١ ــ ٨) يوضح خريطة قاعدة أسماء البيانات السابق شرحها .



شكل (١ ــ ٨)

(١ ــ ٩) التعليقات : Remarks

من الممكن إضافة أية ملاحظات أو تعليقات للبرنامج باستخدام علامة خاصة فتبدأ الملاحظة بالقوس الأيسر } وتنتهى بالقوس الأيمن { ووجود هذه العلامة الخاصة يجعل الكومبيوتر لا يلتفت لما بعدها من ملاحظات حتى القوس

الأيمن الذى تكتمل به الملاحظة .

وكتابة التعليقات فى البرنامج له أهمية كبيرة لا سيما عندما ترغب فى تعديل برنامج مضى وقت طويل على كتابته فإنه بالاستعانة بالملاحظات المكتوبة داخل البرنامج يمكنك تذكر بعض الخطوات بسرعة . كما أنها تفيد فى النواحى التعليمية .

وفى أجهزة الكومبيوتر التى ليس بها هذا النوع من الأقواس يمكنك استخدام العلامتين *) بدلاً من القوس الأيسر والعلامتين (* بدلاً من القوس الأيمن .

ولو كانت الحروف العربية متوفرة فى جهاز الكومبيوتر فلا بأس من كتابة التعليقات باللغة العربية .



■ تمارين على الباب الأول :

س (١ — ١) أذكر أى من أسماء البيانات التالية جائز وأيها غير جائز مع بيان السبب :

(a) a	(f) VAR
(b) datafile1	(g) code number
(c) xrolp5z	(h) real
(d) xi/9	(i) datafile2
(e) passmark	(j) 7 – up

س (١ — ٢) فى المثال السابق كلمات جائزة كأسماء للبيانات ولكنها لا توصى بها . استخرج هذه الكلمات وعلّق عليها من وجهة نظرك .



الباب الثانى

الأنمـــاط والتعـــبيرات
Types & Expressions

مفتتح

تندرج الأدوات المستخدمة فى البرنامج (البيانات والمتغيرات والثوابت المسماه) تحت نوعيات مختلفة هى الأنماط . ومن مستلزمات برنامج باسكال هى الأنماط . ومن مستلزمات برنامج باسكال أن نعلن فى بدايته عن أنماط كل ما نريد استخدامه من أدوات حتى يعد الإجراءات المناسبة للتعامل مع هذه الأدوات وفقاً لأنماطها المختلفة .

ومن الثوابت والمتغيرات والثوابت المسمّاه تتألف التعبيرات التى تندرج بدورها تحت أنماط مختلفة .

وفى هذا الباب نقدم لك كيفية تنظيم أدواتك المستخدمة فى البرنامج وكيفية بناء التعبيرات المختلفة باستخدام هذه الأدوات .

(٢ ــ ١) الأنماط القياسية (Standard Types) :

ولا تنتمى البيانات جميعاً إلى نوعية واحدة فالأسماء تختلف عن الأعداد والأعداد منها الصحيح ومنها الكسر . والكومبيوتر يتعامل مع كل نوعية بطريقة مختلفة لذلك فإن لغة باسكال تحتوى على نوعيات قياسية من البيانات وأسمائها يطلق عليها **الأنماط القياسية (Standard types)** . وبجانب ذلك فإن اللغة تسمح أيضاً بابتكار **أنماط خاصة** بالمبرمج كما سنرى فى الفصول المتقدمة للكتاب . أما الأنماط القياسية فيهى أربعة :

(١) النمط الصحيح (integer) :

وهو النمط الذى تنتمى إليه الأعداد الصحيحة التى لا تحتوى على كسور أو علامة عشرية .

(٢) النمط الحقيقى (real) :

وهو النمط الذى تنتمى إليه الأعداد المحتوية على علامة عشرية أو كسور عشرية .

(٣) نمط اللبنة (char) :

وينتمى إلى هذا النمط كل اللبنات التى تحتوى عليها لغة باسكال وهى الحروف الأبجدية والأرقام من 0 إلى 9 والعلامات الخاصة بأنواعها .

(٤) الأنماط المنطقية (البوليانية)(Boolean) :

وهى تمثل حالتى المنطق :

صحيح	(true)
أو غير صحيح	(false)

(٢ — ٢) *الثوابت (Constants)* :

الثوابت هى البيانات التى نتعامل معها مثل أسماء الأشخاص أو الأعداد الدالة على الأجر والمخزون والرصيد أو أرقام التليفونات وهى تندرج تحت أحد الأنماط الآتية :

(١) الثوابت الصحيحة (Integer constants) :

مثل الأعداد : 125 ، 22 − ، 1281

واستخدام علامة الموجب مع الثوابت اختيارى .

(٢) الثوابت الحقيقية (Real constants) :

وهى قد تكتب بالطريقة المعتادة مثل :

−22.5 ، 0.233 ، 4.0

أو قد تكتب باستخدام الطريقة الأسيّة مثل :

4.5E+6 ، 5.2E−7

ملاحظة :

يعتبر العدد **5.2E − 7** مساوياً للعدد 5.2×10^{-7} .

كما يعتبر العدد **4.5E − 6** مساوياً للعدد 4.5×10^{6} .

(٣) ثوابت اللبنات (Char constants) :

مثل 'A' ، 'B' ، '?'

(٤) الثوابت المنطقية (Boolean constants) :

وهى rue و alse

(٥) ثوابت الحرفيات (string constants) :

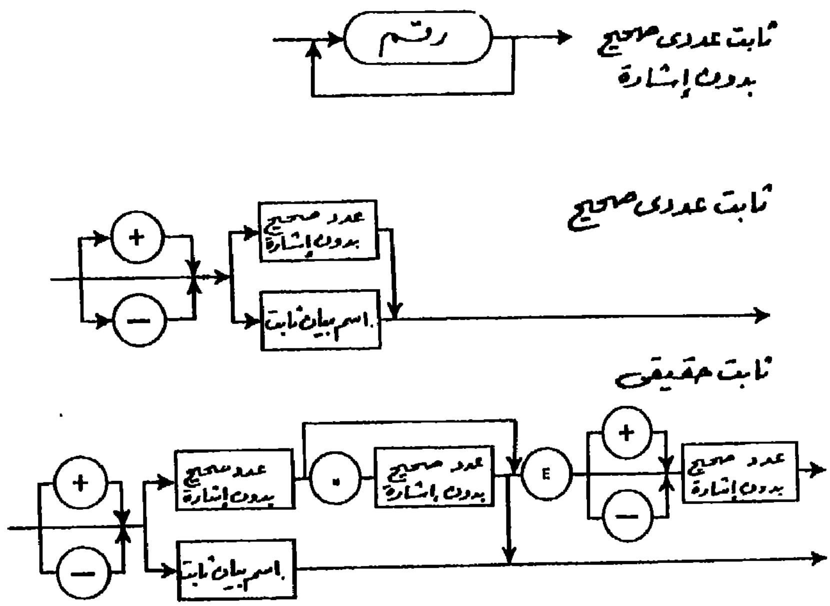
وعلاوة على الأنماط السابقة (القياسية) فإنه يمكن ضم مجموعة من اللبنات

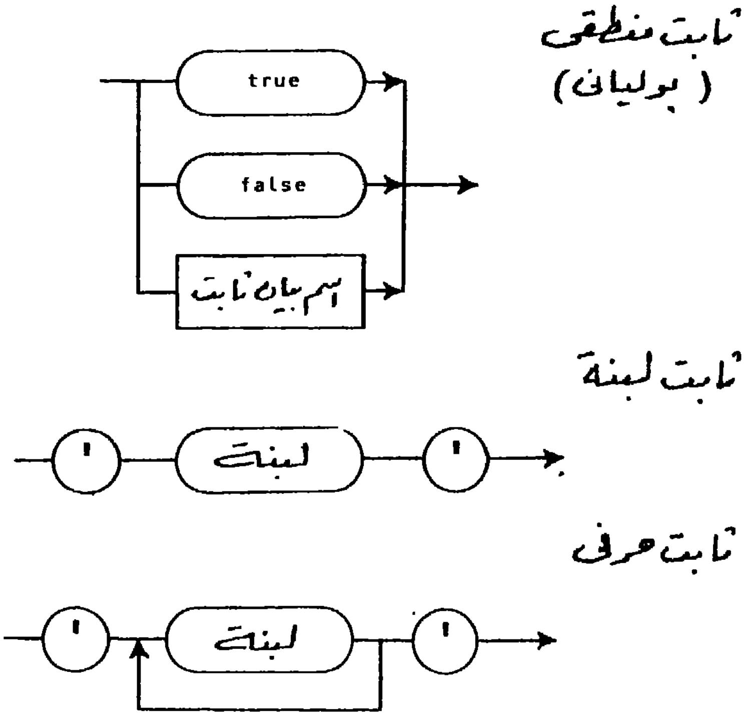
معاً لتكوين ثابت حرفى (string) وقد يطلق عليها أيضاً **مصفوفة اللبنات المُحزّمة** (وسيأتى الحديث عنها فى الأجزاء المتقدمة) .

وأمثلة ثوابت الحرفيات هى :

'ABC' , 'Alexandria' , 'Hazem'

والشكل التالى (٢ ــ ١) يوضح قاعدة تكوين الثوابت أو أسمائها وفقاً للأنماط المختلفة باستخدام خرائط القواعد .





شكل (٢ ــ ١)

(٢ ــ ٣) الإعـــلان *(Declaration)* :

عرفنا أن أسماء البيانات (identifiers) تستخدم لتسمية الأوعية التى تحتوى على البيانات . والواقع أن هناك نوعين من البيانات أحدهما ثابت طوال البرنامج والآخر يتغير أثناء تنفيذ البرنامج .

فعلى سبيل المثال إذا كنا بصدد الحديث عن مساحة الدائرة فإن النسبة التقريبية **TT** **(pi)** تكون دائماً 3.14159 (تقريباً) .

أما القيم الأخرى مثل مساحة الدائرة نفسها فهى تتغير حسب العملية الجارية بالبرنامج .

ويمكن **الإعلان** فى البرنامج عن نوعية أسماء البيانات المزمع استخدامها لتكون إما ثابتة (CONST) أو متغيّرة (VAR) . ويطلق على أسماء البيانات الثابتة اسم الثوابت المسمّاه (named -constants) . وهى نوعية من أسماء البيانات لا تتوفر بكل اللغات ، ومن درس لغة فورتران يجد ما يناظرها تحت اسم البارامترات (parameters) .

(٢ ــ ٣ ــ ١) إعلان الثوابت المُسمّاه (أسماء البيانات الثابتة) CONST

يمكن الإعلان عن أسماء البيانات الثابتة أو الثوابت المسمّاه (named-constants) فى البرنامج كالآتى :

```
CONST Pi = 3.141159;
```

بهذا التعبير يمكن أن يفهم المترجم أن اسم البيان **Pi** سوف يظل ثابتاً طوال البرنامج وأنه من النوع الحقيقى لاحتوائه على علامة عشرية . فإذا ذكر الاسم **Pi** سوف يظل ثابتاً طوال البرنامج وأنه من النوع الحقيقى لاحتوائه على علامة عشرية . فإذا ذكر الاسم **Pi** فيما بعد أثناء البرنامج فإن المترجم يقوم باستبداله تلقائياً بالعدد 3.141159 ويجوز الإعلان عن ثابت مسمى يحتوى على ثابت حرفى ولكن يلزم استخدام علامات الاقتباس فى هذه الحالة مثل :

```
CONST line = '*************';
```

(٢ ــ ٣ ــ ٢) إعلان المتغيرات (أسماء البيانات المتغيرة) VAR

أما الإعلان عن المتغيرات فيتم باستخدام الكلمة VAR كالمثال الآتى :

```
VAR temp :real;
```

الاسم ← temp ، النمط ← real

ويتكون الإعلان عن المتغيرات من الكلمة VAR متبوعة باسم المتغيّر ثم العلامة (:) يليها نمط المتغيّر المعلن عنه . والمتغيرات تنتمى إلى الأنماط القياسية الأربعة السابقة علاوة على نمط مصفوفة الحرفيات المُحّزمة (packed array of

أو نمط الحرفيات (string) وهذا الأخير قد لا تتوفر بكل الطرازات . (char
والإعلان عن المتغيّر لا يتضمن منحه أية قيمة فعلية فهو حتى هذه النقطة فارغ بلا محتويات . ويجوز الإعلان عن عدة متغيرات بعبارة واحدة مع فصل المتغيرات عن بعضها البعض بفاصلة .

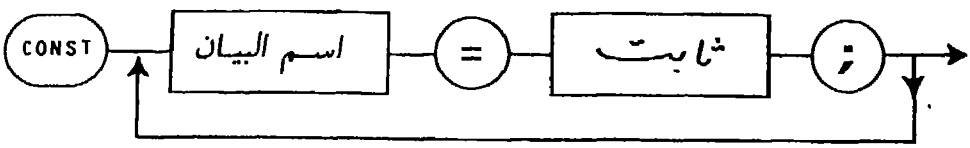
مثال (٢ ـــ ١) :

```
CONST Pi = 3.14159;
VAR raduis, area : real;
```

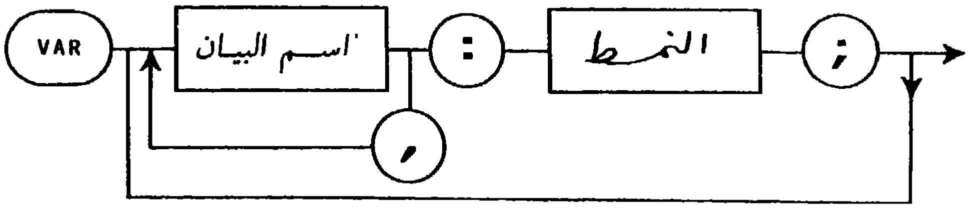
ومما يجدر بالملاحظة حتى الآن هو استخدام العلامات الخاصة (:) ، (=) .

- **فالعلامة** (:) تستخدم للإعلان عن **نمط** اسم بيان .
- **والعلامة** (=) تستخدم **لتحديد قيمة** ثابت مسمّى .

ويوضح الشكلان (٢ ـــ ٢) ، (٢ ـــ ٣) قاعدتي الإعلان عن المتغيرات **(VAR)** والثوابت المسماه (const) بالخرائط .



شكل (٢ ـــ ٢)
الإعلان عن أسماء البيانات الثابتة (أو الثوابت المسماه)



شكل (٢ ــ ٣)

الإعلان عن أسماء البيانات المتغيرة (المتغيرات)

مثال (٢ ــ ٢) :

هذه شريحة من برنامج تستخدم للإعلان عن بعض المتغيرات VAR والثوابت المسماه CONST :

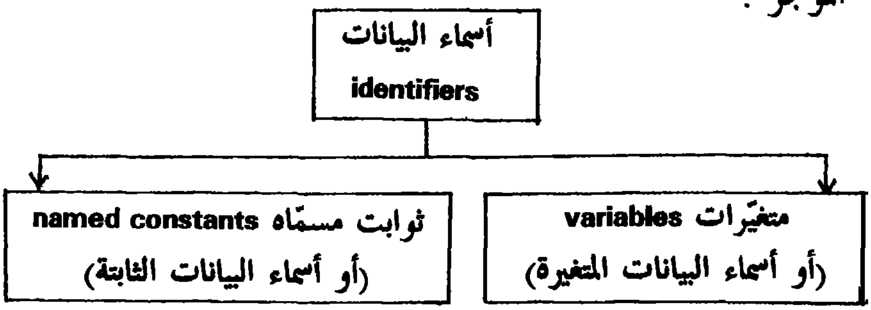
```
CONST    pi = 3.14159; (* type real *)
         zero = 0; (* type integer *)
         vrai = true; (* type boolean *)
         space = ' '; (* type char *)
         heading = 'Pascal program';

VAR      radius : real;
         count : integer;
         firsttime : boolean;
         letter : char;
```

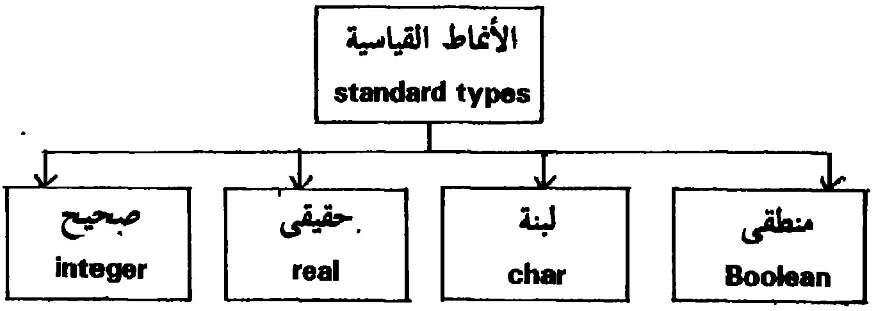
ثوابت مسماه ــ تعليقات ــ متغيرات

شكل (٢ ــ ٤)

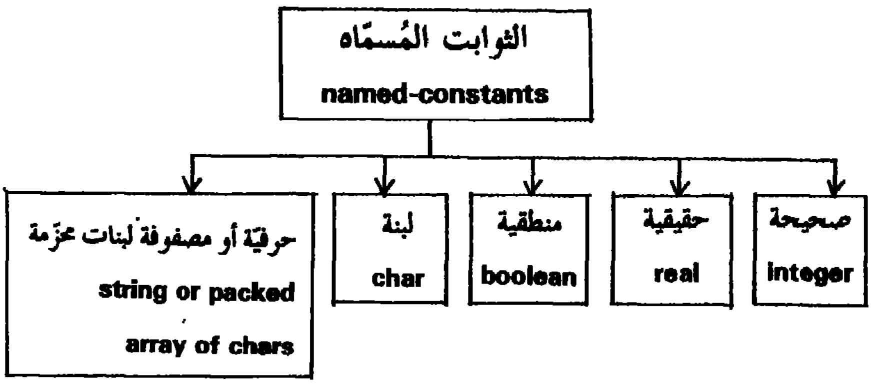
الموجز :



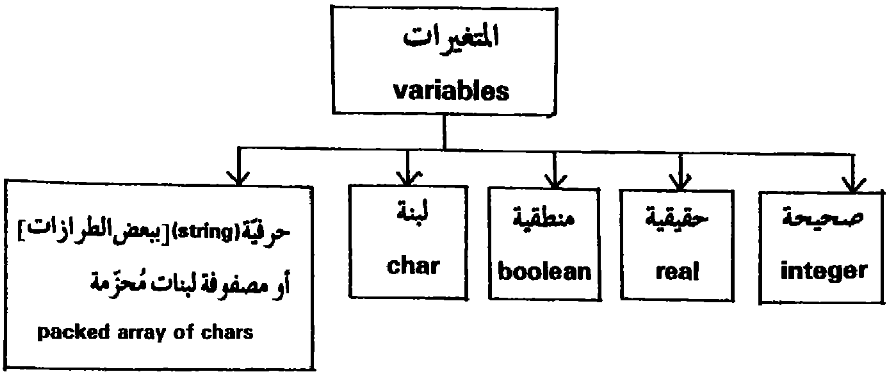
شكل (٢ — ٥)



شكل (٢ — ٦)



شكل (٢ — ٧)

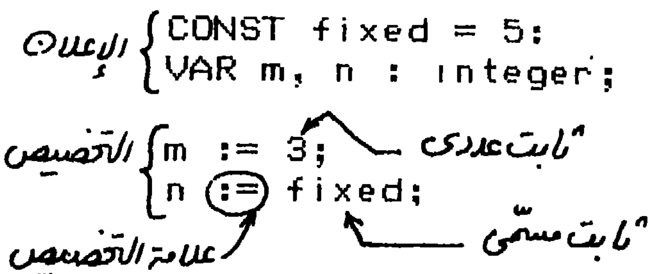


شكل (٢ ــ ٨)

(٢ ــ ٤) *التخصيص (Assignment)* :

تعتبر المتغيرات أوعية فارغة حتى يتم « تعبئتها » بالبيانات . وأحد وسائل « التعبئة » هى عملية التخصيص التى يمكن التعبير عنها كما فى المثال التالى شكل (٢ ــ ٩) :

مثال (٢ ــ ٣) :



شكل (٢ ــ ٩)

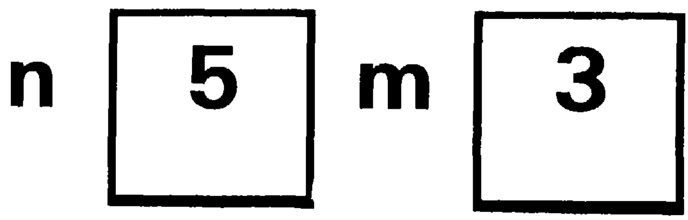
فى هذه الشريحة من برنامج باسكال يعبّر السطر الأول عن الإعلان عن ثابت مسمّى (CONST) أطلقنا عليه الاسم fixed وحددّنا له القيمة 5 .

أما فى السطر الثانى فقد تم الإعلان عن متغيرين (VAR) هما n , m وكلاهما من النوع الصحيح (أو النمط integer) .

أما السطرين الثالث والرابع فقد استخدمناهما للتخصيص للمتغيرات n , m وقد ظهرت بهما علامة جديدة هى علامة التخصيص (=:) .

فى السطر الثالث قد خصصّنا العدد 3 للمتغير m وفى السطر الرابع خصصنا الثابت المسمى fixed للمتغير n وحيث أن قيمة الثابت المسمى هى 5 فإن القيمة المختزنة فى المتغير n قد أصبحت 5 .

ولو مثلنا المتغيرين m ، n بوعائين فى الذاكرة فإن نتيجة عملية التخصيص السابقة سوف تؤدى إلى اختزان العدد 3 فى الوعاء m والعدد 5 فى الوعاء n . شكل (٢ — ١٠) .



شكل (٢ — ١٠)

وللتأكد من النتيجة السابقة يمكن استكمال شكل البرنامج وطباعة محتويات المتغيرين m , n كما فى شكل (٢ — ١١) .

كما يجوز **نقل** محتويات أحد المتغيرات إلى متغير آخر بواسطة عملية التخصيص أيضاً ، وتسمى هذه العملية بتخصيص متغيّر لمتغيّر . ونحن فى غنى

عن القول بأن المتغير الذى يتم تخصيصه (الطرف الأيمن) يجب أن يكون معلوم القيمة أى سبق التخصيص له .

وفى شكل (٢ ــ ١٢) أضفنا علامة تخصيص جديدة هى :

m := n;

معنى ذلك أننا نقلنا محتويات الوعاء n (وهى 5) إلى الوعاء m وبذلك نرى أنه عند طبع محتويات المتغيرات بالبرنامج قد أصبح كل من m ، n يحتوى القيمة 5 .

ويجوز أيضاً تخصيص **تعبير (expression)** صحيح أو حقيقى أو منطقى لمتغيّر ، ولكن مهلاً فالتعبيرات موضوع مستقل سوف نوّفيه بحثاً فى الفقرات التالية .

```
CONST fixed = 5;
VAR m, n : integer;

BEGIN

m := 3;
n := fixed;

writeln('m=',m,' n=',n)

END.

Running
m=3 n=5

>
```

شكل (٢ ــ ١١)

البرنامج

```
CONST fixed = 5;
VAR m, n : integer;

BEGIN

m := 3;
n := fixed;
m := n;      ← تخصيص متغير لمتغير

writeln('m=',m,' n=',n)

END.

Running
m=5 n=5      ← التنفيذ
```

شكل (٢ ــ ١٢)

(٢ ــ ٥) التعبيرات والمؤثرات *Expressions & operators* :

التعبير هو عبارة عن علاقة مركبة بين **الثوابت أو المتغيرات أو الدوال** أو مجموعة منهم ويتم تكوين هذه العلاقة باستخدام المؤثرات .

فعند جمع ثابتين مثل 2,5 نكتب هذه العلاقة فى صورة تعبير كالآتى :

2+5

وبدلاً من جمع الثوابت (الأعداد) يجوز أن نجمع المتغيرات مثل :

a+b

أو نجمع ثابتاً ومتغيراً معاً مثل :

a+5

كل هذه التعبيرات تسمى تعبيرات حسابية لأن المؤثر الذى ساهم فى بناء التعبير هو المؤثر الحسابى (+) .

وعلى ذلك نتوقع أن نوع المؤثرات المستخدمة هو الذى يحدد نوع التعبير الناتج .

ولنبدأ بالتعبيرات الحسابية .

(٢ ــ ٥ ــ ١) التعبيرات الحسابية (الصحيحة والحقيقية) :

لكى نبنى تعبيراً حسابياً ــ سواء كان من النوع الصحيح أو الحقيقى ــ فإننا نستخدم المؤثرات الحسابية التى نوجزها فيما يلى :

المؤثرات الحسابية	التأثير
+	للجمع (وعلامة الموجب)
−	للطرح (وعلامة السالب)
*	للضرب
/	لقسمة الحقيقية
	(مثال :
	ناتج القسمة $\frac{9}{4}$ هو 2.25)

وبالطبع فإن هذه المؤثرات تستخدم وفقاً لأولويات معينة فى التنفيذ كالآتى :

(١) الأقواس أولاً .

(٢) عمليات القسمة والضرب (أيهما يأتى أولاً من اليسار إلى اليمين)

(ملاحظة : المؤثران DIM ، MOD، لهما نفس الأولوية — سيأتى شرحهما) .

(٣) عمليتا الجمع والطرح (أيهما يأتى أولاً من اليسار إلى اليمين) .

مثال (١) : إذا أعطيت التعبير : $\frac{a + b}{c + d}$

فإذا يمكن صياغته بلغة باسكال بالصورة الآتية :

(A + B)/(C + D)

فإذا أهملنا الأقواس أى كتبناه بالصورة :

A + B / C + D

فإن هذا يكافىء التعبير الرياضى :

$$a + \frac{b}{c} + d$$

وهو يختلف تماماً عن الصيغة الأصلية .

وبصفة عامة يمكن القول بأن استخدام الأقواس يجنبك الخطأ أى أن استخدام الأقواس من باب الاحتياط لا يضر بينما يتسبب نقص الأقواس فى أخطاء رياضية جسيمة .

مثال (٢) :

● ونذكر أيضاً المثال الآتى الذى يحتوى على أعداد يمكن حسابها :

```
4 + 3 * 2 - 6 / 2   = 7
  ↓   ↓   ↓   ↓
  2   1   4   3   ←  أولوية التنفيذ
```

● والمثال الآتى تستخدم فيه الأقواس فتأخذ الأولوية الأولى :

```
(4 + 3) * 2 - 6 / 2  = 11
   ↓    ↓   ↓   ↓
   1    2   4   3   ←  أولوية التنفيذ
```

● أنظر أيضاً هذا المثال عندما تتغير أماكن الأقواس :

```
(4 + 3 * 2 - 6) / 2  = 2
   ↓   ↓   ↓    ↓
   2   1   3    4   ←  أولوية التنفيذ
```

● وإذا استخدمنا عدة أقواس متداخلة فإن الأقواس الداخلية تفّك أولاً :

```
((4 + 3) * 2 - 6) / 2  = 4
    ↓    ↓   ↓    ↓
    1    2   3    4   ←  أولوية التنفيذ
```

● **عند احتواء التعبير الرياضى على أقواس زائدة لا تتغير النتيجة كالآتى :**

4 + (3 * 2) − (6 / 2) = 7

● **المؤثرات الخاصة :**

بخلاف المؤثرات السابقة فإن هناك المؤثرين DIV ، MOD اللذين يستخدمان مع الأعداد الصحيحة .

(١) **المؤثر DIV** ويسمى مؤثر القسمة الصحيحة لأن ناتج القسمة يحذف منه الجزء الكسرى .

مثال : ناتج القسمة **4 DIV 19** هو **4**

(٢) **مؤثر الباقى MOD :** ويعطى هذا المؤثر باقى قسمة عدد صحيح على عدد صحيح (وهو اختصار كلمة modulo) .

مثال : التعبير **19 MOD 4** يعطى القيمة 3

أى أن القانون العام لهذا المؤثر هو :

a MOD b = a − (a DIV b) * b

وتأثير هذا المؤثر على القيم السالبة يعتمد على طراز اللغة .

والمؤثران DIV ، MOD لهما نفس الأولوية كمؤثرات القسمة الحقيقية والضرب .

ملاحظة :

حيث أن المؤثرات DIV ، MOD تستخدم مع الأعداد الصحيحة فقط لذلك يمكن تغيير النمط من « صحيح » إلى « حقيقى » متى دعت الحاجة إلى ذلك وذلك بتخصيص محتويات المتغير الصحيح لآخر حقيقى كالمثال الآتى شكل (٢ ــ ١٣) .

مثال (٢ — ٤) :

البرنامج

```
VAR a:real:
    b:integer;

BEGIN

    b:=3:

    a:=b;
    writeln('a=',a,' b=',b)
END.

>
Running

a=   3.0000000000E+00  b=3
```

تخصيص متغير صحيح لآخر حقيقي

التنفيذ — عدد حقيقي — عدد صحيح

شكل (٢ — ١٣)

ولكن لا يجوز تغيير النمط من « حقيقي » إلى « صحيح » حيث أن ذلك يتسبب فى فقد البيانات . وإذا دعت الحاجة إلى ذلك يتم استخدام « دوال التحويل » التى نعرضها فى الفقرة التالية .

مثال (٢ — ٥) حساب مساحة الحجرة :

وهذا المثال يعالج مرة أخرى « مساحة الحجرة » التى عرضناها فى الباب الأول ولكن مع الاستفادة من المؤثرات الحسابية التى عرضناها بإدخال متغيرات جديدة هى :

perimeter	المحيط (ضعف مجموع الطول والعرض)
floorarea	مساحة الأرضية (الطول فى العرض)

هذا علاوة على المتغيرات الأصلية وهى :

الطول length ، العرض width ، الارتفاع height .

ويتم إدخال قيمة هذه المتغيرات أثناء تشغيل البرنامج حيث تتم قراءتها بواسطة العبارة **read** وتخزينها فى الذاكرة (سيأتى شرحها) .

والبرنامج يحسب ويطبع كلاً من المحيط ، مساحة الحوائط ، مساحة الأرضية ، حجم الحجرة .

```
program roomsize3(input,output);

var length, width, height,
    perimeter, floorarea : integer;

begin

   writeln('type in length, width and height');
   read(length, width, height);

   perimeter := 2*(length + width);
   floorarea := length*width;
   writeln('perimeter is ', perimeter);
   writeln('wall area is ', perimeter*height);
   writeln('floorarea is ', floorarea);
   writeln('room volume is ', floorarea*height)

end.
```

شكل (٢ ــ ١٤)

(٢ ــ ٥ ــ ٢) الدوال القياسية (Standard functions) :

الدوال بصفة عامة هى برامج جاهزة توجد ضمن مترجم اللغة وتؤدى وظائف معينة يكثر الحاجة إليها فى التطبيقات المختلفة ولكل دالة اسم خاص تستدعى به عند الحاجة إليها فى البرنامج .

والدالة تتكون من اسم الدالة ، ودليل الدالة (أو البارامتر) كالمثال الاتى :

الاسم trunc(a) الدليل

فهذه الدالة اسمها trunc وهو اختصار كلمة (truncation) بمعنى « قطع » ويطلق عليها أيضاً دالة القطع أو دالة قطع الكسور ، وهى تستخدم لحذف الكسور من الأعداد الحقيقية .

أما الحرف a فهو دليل الدالة (argument) وهو يمثل العدد الذى تؤثر عليه الدالة . ولنر معاً المثال الآتى حيث نقطع العدد 5.566 بالدالة **trunc** .

مثال (٢ ــ ٦) :

البرنامج

```
VAR a:real;

begin
a:=5.566;      <--- العدد الأصلى
writeln(trunc(a));
END.
```

التنفيذ

```
Running
5  <----------- العدد المقطوع
```

شكل (٢ ــ ١٥)

والقطع معناه حذف الكسر تماماً وليس التقريب . أما التقريب فنستخدم له الدالة round كالمثال الآتى :

مثال (٢ ــ ٧) :

```
VAR a:real;

begin
a:=5.566;
writeln(round(a))
END.

Running
6
```

شكل (٢ ـــ ١٦)

وكما نرى فإن العدد 5.566 عندما أثرت عليه دالة التقريب فإنه قد تم تقريبه إلى أقرب رقم صحيح فأصبح 6 .

والشروط التى يجب توفرها فى الدليل لهاتين الدالتين أن يكون من **النوع الحقيقى** وقد يكون ثابتاً أو متغيراً أو تعبيراً .

أما الناتج فهو عدد صحيح ويجوز تخصيصه لمتغير صحيح كالآتى :

b: = tranc(a);

b: = round(a);

حيث : a متغير أو تعبير حقيقى .

b متغير صحيح .

وقد يطلق اسم **دوال التحويل (conversion functions)** على كل من الدالتين trunc ، round لأنهما تستخدمان فى التحويل من حقيقى إلى صحيح .

● كما تمد لغة باسكال بالدوال القياسية الآتية :

الدالة	الوظيفة
sqr(i)	تعطى مربع الدليل i سواء كان حقيقياً أو صحيحاً
abs(i)	تعطى القيمة المطلقة للدليل i سواء كان حقيقياً أو صحيحاً

أمثلة :

sqr(3)	تعطى	9
sqr(2.5)	تعطى	6.25
abs(−25)	تعطى	25
abs(2)	تعطى	2
abs(−27.12)	تعطى	27.12

● أما المجموعة الآتية من الدوال فهى تعطى القيمة الحقيقية وتؤثر على أدلة حقيقية فقط :

الدالة	الوظيفة
sqrt(r)	دالة الجذر التربيعى للدليل r
sin(r)	دالة جيب الزاوية r (r بالتقدير الدائرى) .
cos(r)	دالة جيب تمام الزاوية r (r بالتقدير الدائرى)
arctan(x)	الزاوية التى ظلها x (الجواب بالتقدير الدائرى) (فى طراز اللغة UCSD تسمى هذه الدالة atan)
exp(x)	الدالة الأسية للدليل x (e^x)
ln(x)	اللوغاريتم الطبيعى للدليل x

(٢ ـ ٥ ـ ٣) المؤثرات الأسيّة (exponential operators) :

لا يوجد فى لغة باسكال المؤثر الذى يرفع إلى أس كما فى لغتى بيسك (↑) وفورتران (**) . حيث أن هذا المؤثر يحتل زمناً ملموساً من وقت الكومبيوتر . ومع ذلك يمكن الاستعاضة عنه بدالة التربيع sqr .

والبرنامج التالى يوضح كيف يمكن استخدام هذه الدالة للحصول على المتغير x مرفوعاً لأسس مختلفة 2 ، 3 ، 4 . وتظهر لنا عبارة القراءة **read** فى السطر الأول بعد بداية البرنامج والتى تستخدم لمنح المتغير x قيمة عددية يستقبلها الكومبيوتر أثناء التنفيذ (وسوف نتحدث عنها بالتفصيل فى حينه) .

مثال (٢ ـ ٨) :

البرنامج :

```
program power(input,output);
     var x:integer;
begin
     read (x); ◄─── استقبل قيمة المتغير x
     writeln;
     writeln('if x= ',x,' then: ');
     writeln(' x'2= ',sqr(x));
     writeln(' x^3= ',x*sqr(x));
     writeln(' x^4= ',sqr(sqr(x)))

end.
```

التنفيذ :

```
2  ◄─── القيمة المدخلة للمتغير x
if x= 2 then:
 x^2= 4  }
 x^3= 8  }  ◄─── النتائج
 x^4= 16 }
```

شكل (٢ ـ ١٧)

ولعلنا نلاحظ أن دليل الدالة sqr يمكن أن يكون عبارة عن دالة أخرى وهذا ينطبق على سائر الدوال .

وبالإضافة إلى هذه الدالة فتوجد أيضاً دوال أخرى تفيد فى الرفع إلى أس مثل الدالة اللوغاريتمية **ln** والدالة الاسيّة exp .

ويمكن إيجاد العلاقة **a↑b** (a مرفوعة إلى الأس b) باستخدام التعبير :

exp(ln (a)*b))

مثال (٢ ــ ٩) سوبر ماركت :

مطلوبٌ من عامل الخزينة « الكيس » ، فى محلات « السوبر ماركت » الكبيرة ، أن يكون سريعاً فى إجراء العمليات الحسابية ولكن مهما كانت سرعته فإن الكومبيوتر لا زال يتفوق بسرعته الهائلة فى هذه العمليات . والبرنامج التالى يمكن استخدامه لمساعدة عامل الخزينة حيث يخبره بالمبلغ المتبقى من الجنيه فى صورة عدد من الوحدات النقدية مثل عدد العشرة قروش والخمسة قروش والقروش إلى آخره .

ومدخلات البرنامج هى : سعر البضاعة المشتراه (price) .

```
program change(input,output);

var  price, change, noof50s, noof10s,
     noof5s, noof2s, noof1s : integer;

begin
   read(price);

   change  := 100 - price;
   noof50s := change div 50;
   change  := change mod 50;

   noof10s := change div 10;
   change  := change mod 10;

   noof5s  := change div 5 ;
   change  := change mod 5 ;

   noof2s  := change div 2 ;
   change  := change mod 2 ;

   noof1s  := change;

   writeln('change due is:  no of 50s    ', noof50s)
   writeln('                no of 10s    ', noof10s)
   writeln('                no of  5s    ', noof5s);
   writeln('                no of  2s    ', noof2s);
   writeln('                no of  1s    ', noof1s)

end.
```

شكل (٢ — ١٨)

مثال (٢ — ١٠) التحويل من مئوى إلى فهرنهيت :

هذا البرنامج يقوم بقراءة درجة الحرارة بالتدريج ويحسب ما ينظرها بالتدريج الفهرنهيت وفقاً للعلاقة :

$$\textbf{fdegrees} = \textbf{cdegrees} \times \frac{9}{5} + 32$$

حيث fdegrees هى درجة الحرارة بالفهرنهيت .

cdegrees هى درجة الحرارة المئوية .

ثم يستخدم البرنامج الدالة round فى طبع نتيجة تقريبية للعدد الناتج علاوة على القيمة الصحيحة فتكون إجابته كالمثال الآتى :

21.50c = 70.70f or approx 71f

تقريباً

```
program conversion(input,output);

var cdegrees, fdegrees : real;

begin
   read(cdegrees);
   fdegrees := cdegrees * 9/5 + 32;
   writeln(cdegrees, 'c = ', fdegrees, 'f or approx ',
                          round(fdegrees), 'f')

end.
```

التحويل من مئوى إلى فهرنهيت

شكل (٢ ـــ ١٩)

(٢ ـــ ٦) *التعبيرات المنطقية* *(Boolean Expressions)* :

يعتمد التعبير المنطقى فى تكوينه على المؤثرات « **العلاقية** » و « **المنطقية** » التى سيأتى شرحها وعادة تكون قيمة التعبير المنطقى أحد شيئين : صحيح (True) أو غير صحيح (false) .

وعادة يستخدم التعبير المنطقى للتعبير عن شرط ما قد يتحقق أو لا يتحقق .

(٢ ـــ ٦ ـــ ١) المؤثرات العلاقية (Relational operators) :

تستخدم المؤثرات العلاقية فى المقارنات بين القيم المختلفة للمتغيرات والتعبيرات وهى تشمل الآتى :

المعنى	المؤثر
أكبر من	>
أصغر من	<
أكبر من أو يساوى	>=
أصغر من أو يساوى	<=
يساوى	=
ليس أكبر من	< >

وفى الأمثلة الآتية نناقش استخدام هذه المؤثرات فى التعبيرات المنطقية :

التعبير الآتى يعتبر تعبير منطقياً :

$$(a+b) < (c+d)$$

وعند تقييم هذا التعبير قد يعطى القيمة « صحيح » (true) إذا كان المجموع (a+b) أقل من المجموع (c+d) ، وإلاَّ فإنه يعطى القيمة « غير صحيح » (false) .

ونلاحظ ضرورة وجود الأقواس فى هذا التعبير .

ولنر استخدام المؤثرات العلاقية مع التعبيرات المنطقية فى هذه الشريحة من البرنامج (شكل ٢ ــ ٢٠) :

مثال (٢ ــ ١١) :

```
var     a,b,c,d:real;
        first,
        second,
        same:boolean;
begin
        first:=a>b;
        second:=c<d;
        same:=first=second;
        read(a,b,c,d);
        .
        .
        .
end.
```

شكل (٢ ــ ٢٠)

ففى هذا المثال بعد أن يتم الإعلان عن المتغيرات المنطقية الثلاثة first ، second ، same ، نبدأ فى الشروط التى تمنح هذه المتغيرات القيمة الصحيحة true والتى إذا لم تحقق فإنها تحتوى على القيمة غير الصحيحة false .

فالمتغير first تكون قيمته true إذا كانت (a>b) . والمتغير second تكون قيمته true إذا كانت (c<d) أما المتغير same فتكون قيمته صحيحة إذا كان المتغيران first ، second متساويين . ولنجرب الآن تخصيص بعض الأعداد لهذه المتغيرات ونرى النتيجة شكل (٢ ــ ٢١) :

مثال (٢ ــ ١٢) :

البرنامج :

```
var    a,b,c,d:real;
       first,              ← الإعلان
       second,
       same:boolean;
begin
       first:=a>b;         }
       second:=c<d;        } ← الشروط
       same:=first=second; }
       a:=77 ;b:=60; c:=6; d:=54;   ← التخصيص
       write(first,' ',second,' ',same)
end.                                ↑ الطباعة
```

التنفيذ :

```
TRUE TRUE TRUE
```

شكل (٢ ــ ٢١)

وكما نرى فالنتيجة التى طبعها البرنامج هى :

TRUE TRUE TRUE

وهى تمثل القيم المنطقية التى تحتوى عليها المتغيرات المنطقية الثلاثة . وبالتأمل فى الأعداد التى خصصناها للمتغيرات a,b,c,d نجد أنها جميعاً تحقق الشروط التى تجعل المتغيرات المنطقية صحيحة (true) .

هناك وسيلة أقوى للاستمتاع بهذا البرنامج وفهمه بعمق أكثر .

ماذا لو استخدمنا العبارة **read** لكى نستقبل قيماً مختلفة للمتغيرات a,b,c,d عند كل تنفيذ للبرنامج ؟ إننا بذلك يمكن أن نشاهد كل الاحتمالات المختلفة للنتائج . أنظر البرنامج المعدّل شكل (٢ ــ ٢٢) مصحوباً بالتنفيذ لقيم مختلفة للمتغيرات .

ولنلاحظ عند إدخال المتغيرات أنها تدخل وفقاً لنفس الترتيب الوارد فى العبارة read من اليسار إلى اليمين .

ففى التنفيذ الأخير مثلاً نجد أن a = 4 ، b = 3 ، c = 2 ، d = 1 .

وكذلك عند طبع قيمة المتغيرات المنطقية فإنها تطبع حسب الترتيب الوارد فى عبارة الطبع writeln من اليسار إلى اليمين أيضاً .

مثال (٢ ــ ١٣) :

البرنامج :

```
var    a,b,c,d:real;
       first,
       second,
       same:boolean;
begin
       first:=a>b;
       second:=c<d;
       same:=first=second;
       read(a,b,c,d);
       writeln;
       write(first,' ',second,' ',same)
end.
```

التنفيذ :

القيم المُدخلة للمتغيرات a، b، c، d بالترتيب من اليسار

```
2 2 2 1
TRUE FALSE FALSE
```

النتائج

```
1 1 1 1
FALSE FALSE TRUE

1 2 3 4
FALSE FALSE TRUE

9 7 1 4
TRUE TRUE TRUE

4 3 2 1
TRUE TRUE TRUE
```

شكل (٢ ــ ٢٢)

مثال (٢ ــ ١٤) مقارنة الكميّات الحقيقية :

عند استخدام الكميات الحقيقية يجب أن يضع المبرمج فى اعتباره أنها لا يمكن تمثيلها فى الكومبيوتر بدقة متناهية فجميع القيم الحقيقية تخضع للتقريب .

لذلك فلا معنى لأن نختبر قيمتين من النوع الحقيقى بغرض معرفة ما إذا كانتا متساويتين أم لا .

وبدلاً من ذلك يتم اختبار الفرق بين القيمتين بحيث يكون أصغر من قيمة صغيرة جداً محددة سلفاً (وتختلف هذه القيمة بالطبع بحسب نوع التطبيق) .

ولنر شريحة البرنامج التالية التى تختبر الفرق بين الكميتين الحقيقيتين a,b بحيث يكون أصغر من قيمة السماح (tolerance) المحدّدة بالكسر (1E−3) :

```
CONST      tolerance  =  1.0e-3;
VAR        a,b  :  real;
           same  :  boolean;
....
same  :=  abs(a-b)  <  tolerance:
```

شكل (٢ ــ ٢٣)

(٢ ــ ٦ ــ ٢) المؤثرات المنطقية Boolean Operators :

تستخدم المؤثرات المنطقية (علاوة على المؤثرات العلاقية) لبناء التعبيرات المنطقية .

والمؤثرات المنطقية هى :

مؤثر النفى المنطقى **NOT**
مؤثر الضرب المنطقى **AND**
مؤثر الجمع المنطقى **OR**

ومن كان حديث التعامل مع مثل هذه المؤثرات يمكنه **الرجوع للملحق (هـ) حيث نشرح معناها تفصيلاً** .

أما من ناحية أولوية هذه المؤثرات فإن مؤثر النفى المنطقى **NOT** يتمتع بالأولوية الأولى ويليه مؤثر الضرب **AND** ثم مؤثر الجمع **OR** .

فعلى سبيل المثال يمكننا كتابة التعبير الآتى :

NOT a AND b OR c AND NOT d

حيث a ، b ، c ، d كلها متغيرات منطقية .

فما معنى هذا التعبير ؟ لعلنا نستطيع الاقتراب من معناه أكثر إذا استخدمنا الأقواس لتوضيح الأولويات كالتالى :

((NOT a) AND b) OR (c AND (NOT d))

بهذه الصورة يصبح التعبير أكثر وضوحاً حيث نرى أن الأقواس الداخلية ـــ وهى تتمتع بالأولوية الأولى ـــ قد احتضنت الجزء **(NOT a)** . فأصبح هذا الجزء ذا أولوية أولى . أما المؤثر **AND** بطرفيه اللذين يربطهما ببعض فقد وقع داخل القوس ذى الأولوية الثانية . أما المؤثر **OR** الذى يقع خارج الأقواس فسوف يأتى دوره كأولوية ثالثة .

ومن الجدير بالذكر أن استخدام الأقواس هنا لمجرد التوضيح ومَن يستطيع أن يرى الأولويات بلا أقواس فليستخدم التعبير الأول .

فماذا لو أردنا أن يسرى مفعول المؤثر NOT على كل مكونات التعبير المنطقى ؟ لابد من استخدام الأقواس فى هذه الحالة . ولنر مثالاً آخر :

I:= (a > b) AND firsttime

تأخذ هنا القيمة Trueإذا كان التعبير الذى فى الطرف الأيمن مساوياً للقيمة true (أى أن كل من طرفيه يساوى القيمة true) .

فإذا أردنا أن تأخذ I القيمة true عندما يكون التعبير فى الطرف الأيمن مساوياً القيمة false فإن هذا النفى يلزم معه استخدام المؤثر NOT بالصورة الآتية :

I:= NOT ((a > b) AND firsttime)

هل لاحظت أن التعبير فى الطرف الأيمن قد وقع بالكامل داخل قوسين يسبقهما المؤثر NOT ؟ .

هذا ضرورى لصحة التعبير المقصود ، فمن الخطأ أن ننفى التعبير بلا أقواس كالآتى :

I:= NOT (a > b) AND firsttime

فالتعبير فى الطرف الأيمن هنا له معنى مختلف فالمؤثر NOT فى هذه الحالة يؤثر على العلاقة التى بين القوسين فقط . أى أن هذا التعبير يقول :

« تكون قيمة I هى true إذا تحقق معاً الشرطان الآتيان :

١ ــ أن تكون قيمة التعبير **(a > b)** هى false (بمعنى ألاّ تكون a أكبر من b) .

٢ ــ وأن تكون قيمة المتغير firsttime هى true » .

(٢ ـ ٦ ـ ٣) قواعد « دى مورجان » :

أما العالم « أغسطس دى مورجان » (Augutus de Morgan) فله بصمات واضحة فى علم الجبر البوليانى أو الجبر المنطقى حيث وضع قاعدتين هامتين شملتهما لغة باسكال بالصورة الآتية :

١ ـ التعبير الآتى :

NOT (a OR b)

يُعتبر مكافئاً للتعبير :

NOT a AND NOT b

٢ ـ والتعبير الآتى :

NOT (a AND b)

مكافىء للتعبير :

NOT a OR NOT b

وإثبات هذه القواعد ليس مجال اهتمامنا فى هذا الكتاب . وهى بصفة عامة لا تهم إلاّ المتخصصين والرياضيين .

(٢ ـ ٦ ـ ٤) الدالة odd :

تعتبر الدالة odd من الدوال القياسية وهى تنتج قيمة منطقية إذا أثرت على تعبير صحيح (integer expression) وصيغتها كالآتى :

odd (integer expression)

الدالة — **الدليل**

(تعبير صحيح)

والنتيجة تكون صحيحة true إذا كانت قيمة التعبير عدداً فردياً أما إذا كانت قيمة التعبير عدداً زوجياً تصبح النتيجة false .

فالدالة الآتية تعطى القيمة true :

odd (1)

والدالة الآتية تعطى القيمة false :

odd (128)

ولنر البرنامج شكل (٢ ــ ٢٤) مصحوباً بالتنفيذ حيث يطبع قيم الدالتين السابقتين :

مثال :

البرنامج :

```
begin

writeln (odd(1));
writeln (odd(128));

end.
```

التنفيذ :

```
TRUE
FALSE
```

شكل (٢ ــ ٢٤)

(٢ــ٦ــ٥) مقارنة اللبنات :

لعل من درس لغة بيسك من قبل ينتظر بلهفة أن يرى كيف تتعامل لغة

باسكال مع الحرفيات أو مع اللبنات وكيف تتم العمليات المختلفة على **التعبيرات الحرفية** . والواقع أن لغة باسكال لا تحتوى على مؤثرات حرفية أو مؤثرات على اللبنات (character operators) ومع ذلك فيمكن استخدام اللبنات فى التعبيرات المنطقية كالمثال الآتى :

```
VAR B : boolean;
ch1,ch2 : char;

b := ch1 = 'a',
```

· فما معنى العبارة الأخيرة ؟

إنها تقول بأن المتغير البوليانى b يمكن أن يأخذ القيمة true إذا كان متغير اللبنة ch1 يحتوى على الحرف 'a' .

ويمكن أيضاً إضافة العبارة الآتية :

```
b := ch1 = ch2
```

بمعنى أن المتغير b يأخذ القيمة true إذا تساوت محتويات كل من متغيرى اللبنة ch1 ، ch2 .

ومع ذلك فموضوع معالجة الحرفيّات واللبنات سوف نتعرض له بالتفصيل فى البابين السادس والسابع .

■ تمارين على الباب الثانى :

س (٢ — ١) : اكتب إعلاناً بلغة باسكال للمتغيرات التى تصلح لاحتواء كل نوع من أنواع البيانات التالية :

(أ) درجات امتحان (أعداد صحيحة)

(ب) متوسط درجات عدة امتحانات .

(جـ) شفرة حرفية 'A' أو 'O' لتوضيح مستوى الامتحان .

(د) نتيجة إذا ما كان الطالب قد نجح أو رسب .

ثم اكتب عبارات تخصيص بلغة باسكال تمنح بها القيم الآتية للمتغيرات التى أعلنت عنها فى الفقرة السابقة :

(أ) 51

(ب) 47.5

(جـ) المستوى A

(د) ناجح

س (٢ — ٢) : اكتب قيمة التعبيرات الآتية :

```
16/5
16 div 5
16 mod 5
19/5
19 div 5
19 mod 5
 8 div 3 * 3
 7 + 5 div 3
13 - 5 mod 3
```

س (٢ ـــ ٣) : تقوم محطة رادار جويّة باكتشاف الطائرات المعادية المقتربة من المجال الجوى .

ويعطى جهاز الرادار زاوية الارتفاع (angle of elevation) وكذلك مدى الطائرة (range) فى اتجاه الشعاع الرادارى المرسل .

اكتب إعلاناً بلغة باسكال وكذلك العبارات اللازمة لحساب كل من الإحداثى الأفقى والإحداثى الرأسى الممثل للطائرة على مستوى كارتيزى .

(**ملاحظة** : الدرجة = **00174532925** زاوية نصف قطرية) .

س (٢ ـــ ٤) : إذا أعطيت المسافة بين مدينتين وسرعة السيارة بالميل فى الساعة (mph) وساعة القيام من أحد المدينتين ، اكتب برنامجاً يطبع ساعة الوصول إلى المدينة الأخرى .

يتم إدخال ساعة القيام بالصيغة الآتية :

0845

أو

1357

العدد الأول معناه الساعة الثامنة وخمسة وأربعون دقيقة (صباحاً) .

والعدد الثانى معناه الساعة الواحدة (١٣) وسبعة وخمسون دقيقة ظهراً والمطلوب أن يكون خرج البرنامج على الصورة :

8 – 45

أو

13 – 57

للتعبير عن نفس المواعيد المذكورة .

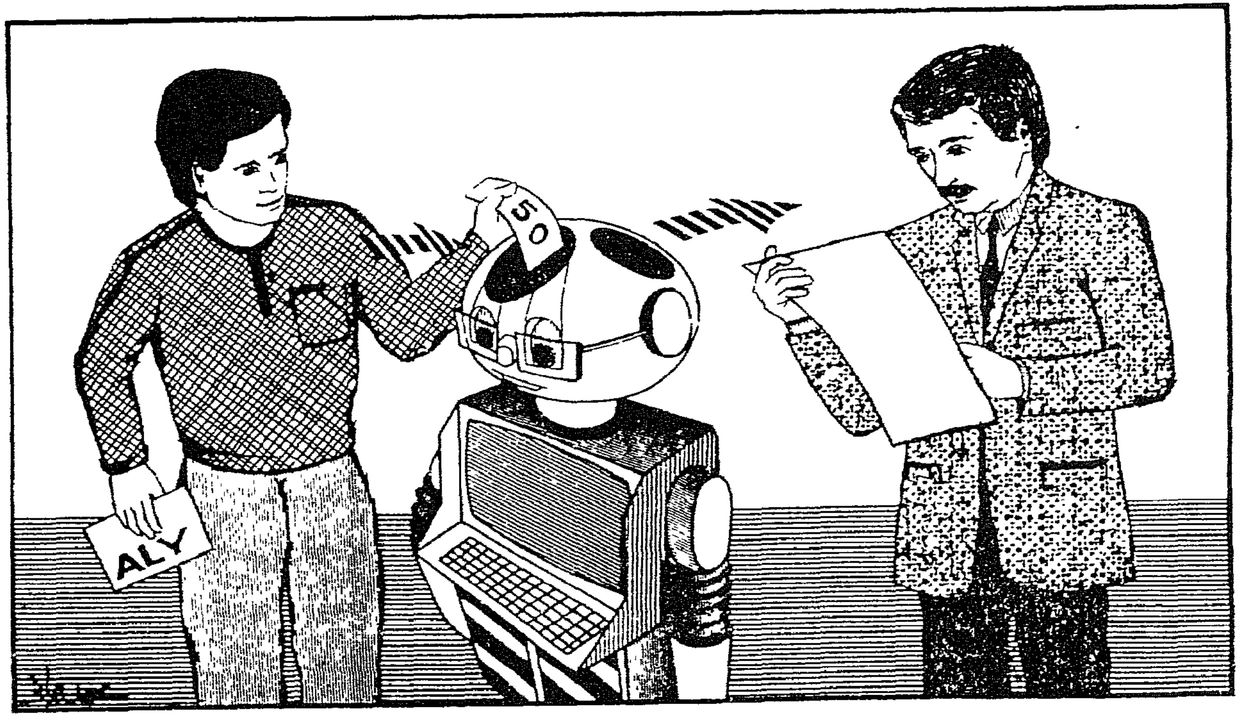
وسوف نفترض أن الرحلة تستغرق يوماً واحداً .

الباب الثالث

إدخال البيانات
طباعة المعلومات
INPUT/OUTPUT

مفتتح

نستعرض فى هذا الباب عبارات لغة باسكال التى نستطيع بها إدخال البيانات عن طريق لوحة الأزرار (أو النهاية الطرفية)، أو إدخالها ضمن البرنامج نفسه. كما نعرض طرق استرجاع المعلومات المختلفة على الشاشة وطباعتها فى الحالتين بالصورة الملائمة.



إدخال البيانات واسترجاع المعلومات

(٣ ـــ ١) إدخال البيانات إلى البرنامج read readln :

نستطيع الآن أن نعود إلى البرنامج الذى عرضناه فى الباب الأول لنتحدث عنه بمزيد من التفصيل .

مثال (٣ ـــ ١) :

```
program roomsize(input,output);

var length, width, height : integer;

begin

   read(length, width, height);
   writeln('your room needs ', length*width,
         ' sq.m. of vinyl');
   writeln('and ', 2*(length + width)*height,
         ' sq.m. of wallcovering')
end.
```

شكل (٣ ـــ ١)

فى هذا البرنامج أعلنّا عن ٣ متغيرات هى width , height , length كمتغيرات عددية صحيحة فى فقرة الإعلانات .

فإذا انتقلنا إلى العبارة الأولى فى الجزء التنفيذى من البرنامج والتى تلى الأمر BEGIN مباشرة سوف نلتقى بالعبارة **read** بمعنى (اقرأ) يليها أسماء المتغيرات الثلاثة بين قوسين .

عند تنفيذ هذه العبارة فإن الكومبيوتر يتوقف منتظراً من مستخدم البرنامج أن يُدخل إليه ثلاثة أرقام عن طريق لوحة الأزرار وعندما يستقبل الكومبيوتر

الأرقام الثلاثة فإنه يخصص للمتغيرات الثلاثة length , wridth , height بحسب ترتيب دخولها وبحسب ترتيب أسماء المتغيرات فى العبارة read .

فالكومبيوتر بالطبع لا يعرف ما معنى length ؟ أو width ولكنه يعرف فقط أنه سيستقبل ٣ قيم لمتغيرات ثلاثة يخزنها فى أماكنها فى الذاكرة كل فى وعاء معيّن كما فى الشكل وهو يستخدم أسماء البيانات كعناوين يستدل بها على أوعية البيانات .

فالرقم الأول الذى ندخله سوف يخصصه الكومبيوتر للمتغير lenght لأنه هو المتغير الأول فى العبارة read والرقم الثانى سيكون هو width والثالث هو height .

3	**2**	**3**	
length	**width**	**height**	

شكل (٣ ـــ ٢)

وبمجرد أن تمتلىء أوعية المتغيرات الثلاثة بالبيانات يبدأ الكومبيوتر فى إجراء عمليات المعالجة اللازمة وهى حساب قيمة التعبيرات الحسابية الواردة فى عبارات الطبع وهى :

length*width

2*(length + width)*height

ونلاحظ **ظهور العبارة** (input) فى مستهل البرنامج دلالة على أن هناك عملية إدخال بيانات بهذا البرنامج وعلى الكومبيوتر أن يعد العدّة لذلك .

وعندما ندخل بيانات عددية للكومبيوتر يجب أن تكون مفصولة عن بعضها البعض بمسافة خالية (أو أكثر) حتى يستطيع التمييز بينها (وهذا لا ينطبق على اللبنات كما سنرى فعند تنفيذ هذا البرنامج سوف نلتقى بعلامة استفهام فى أقصى يسار الشاشة وعلينا أن ندخل الأرقام المطلوبة كالمثال الآتى :

?3 2 3

إذا كانت هذه هى الأرقام المدخلة فسوف يوافينا البرنامج بالرد الآتى فوراً :

your room needs 6 sq.m. of vinyl
and 30 sq.m. of wallcovering

كما يمكن استخدام العبارة **readln** بدلاً من العبارة **read** ، والفارق بينهما هو أن العبارة readln تسمح بالانتقال إلى سطر جديد بعد إدخال البيانات وهى تغنى عن استخدام العبارة writeln للانتقال إلى سطر جديد .

وبالطبع يمكن تنفيذ البرنامج أكثر من مرة ببيانات جديدة فى كل مرة ، كالمثال الآتى :

?5 3 3 ← (البيانات المدخلة)
your room needs 15 sq.m. of vinyl
and 48 sq.m. of wallcovering ← (إجابة الكومبيوتر)

ألا ينقص هذا البرنامج شيء ؟

إننا الآن نعرف أن البرنامج عندما يتوقف منتظراً إدخال البيانات على يمين علامة الاستفهام ـــ آنه يريدنا آن ندخل له ثلاثة ارقام متتالية تعبر عن الطول والعرض والارتفاع ولكن لو مرّ على إنشاء البرنامج فترة قصيرة فلا شك أننا سننسى ترتيب البيانات وربما نوعياتها أيضاً . هذا فضلاً عن أن شخصاً آخر لن يستطيع استخدام البرنامج بدون أن يراجع سطور البرنامج نفسه ليعرف ترتيب دخول هذه البيانات . والبرنامج الجيد يجب أن يكون سهل الاستخدام بمعنى أن تكون تعليمات استخدامه مطبوعة على الشاشة بحيث لا يتطلب الأمر قراءة البرنامج نفسه .

فما هى الإضافة المنتظرة التى تجعل البرنامج أكثر وضوحاً عندما يتوقف منتظراً منا إدخال البيانات ؟

(٣ ـــ ٢) *طباعة رسالة عند إدخال البيانات :*

ماذا لو ظهرت علامة الاستفهام مسبوقة ببعض العبارات التوضيحية كالمثال الآتى :

type in length, width, and height ← الرسالة

? 3 2 4 ← البيانات المدخلة

إن العبارة (أو العبارات) التى سبقت علامة الاستفهام والتى نصطلح على تسميتها بالرسالة (message) تدل المستخدم على نوعية وترتيب البيانات التى ينتظرها الكومبيوتر ففى هذه الحالة سوف يفهم المستخدم أن الكومبيوتر ينتظر إدخال ثلاثة أعداد متتابعة تمثل الطول والعرض

والارتفاع بالترتيب .

كيف يمكن تحقيق ذلك .

يمكن إضافة عبارة لطباعة هذه الرسالة قبل عبارة الدخل (read) مباشرة ، هذه العبارة هى :

writeln('type in length, width, and height');

بذلك يكون البرنامج فى صورته الكاملة كالآتى :

مثال (٣ ــ ٢) :

```
program roomsize2(input,output);

var length, width, height : integer;

begin

   writeln('type in length, width and height');
   read(length, width, height);

   writeln('your room needs ', length*width,
          ' sq.m. of vinyl');
   writeln('and ', 2*(length + width)*height,
          ' sq.m. of wall covering')
end.
```

شكل (٣ ــ ٢)

(٣ ــ ٣) طباعة الرسائل والنتائج Writeln :

رأينا فى الفقرة السابقة أن العبارة writeln تستخدم فى طباعة رسالة ما على الشاشة . وأن الرسالة المطلوب طباعتها توضع بين علامتى اقتباس كقاعدة عامة . فكل ما تضعه بين علامتى الاقتباس يظهر مطبوعاً على الشاشة بما يحتويه من فواصل ومسافات خالية .

وقد أشرنا من قبل أن العبارة writeln تعنى طباعة ما يأتى بعدها (بين قوسين) على سطر مستقل .

كما التقينا من قبل بالعبارة writeln فى الباب الأول وعلمنا أن من خصائصها أيضاً إجراء بعض العمليات الحسابية أو تقييم التعبيرات قبل طباعتها وهذا واضح فى السطرين الأخيرين من البرنامج .

فالسطر الأول يقول :

```
writeln('your room needs', length * width,
'sq.m. of vinyl');
```

نرى أن هذه العبارة تحتوى على رسالتين وضعت كل منهما بين علامتى اقتباس وهما :

```
your room needs
sq.m. of vinyl
```

كما تحتوى العبارة على التعبير الحسابى الذى أتى بدون علامات اقتباس :

length*width

وعندما يصادف الكومبيوتر ضمن محتويات الطبع جزءاً بدون علامتى اقتباس ، يتوقع أن يكون تعبيراً حسابياً يحتاج لمعالجة وطريقة مختلفة فى الطباعة ، ولنر هذه الأمثلة :

مثال (٣ ــ ٣) :

العبارة	النتيجة المطبوعة
writeln(25)	25
writeln(3*2)	6
writeln('3 x 2 = ',3*2)	3 x 2 = 6

شكل (٣ ــ ٣)

نخرج من المثال السابق بالملاحظات الآتية :

١ ــ إذا كان البيان المطلوب طبعه بياناً عددياً (مثل ٢٥) أو تعبيراً حسابياً (مثل 2 * 3) فلا يوضع البيان بين علامتى اقتباس والتعبير الحسابى يتم تقييمه

قبل طباعته .

٢ ـــ إذا كان البيان المطلوب طبعه رسالة أو بصفة عامة بياناً حرفيّاً (string) فإن البيان يوضع بين علامتى اقتباس .

٣ ـــ يجوز أن تشتمل عبارة الطبع على بيانات مختلفة (عددية وحرفية) وتفصل البيانات عن بعضها البعض بفصلة .

مثال (٣ ـــ ٤) :

العبارة	النتيجة المطبوعة
writeln(length * width)	**15**
writeln('area is ' length * width)	**area is 15**

فى هذا المثال أخذ التعبير الحسابى صورة مختلفة حيث احتوى على متغيرات بدلاً من الأعداد . ويقوم الكومبيوتر فى هذه الحالة باسترجاع القيمة العددية لهذه المتغيرات من خانات الذاكرة قبل تقييم التعبير الحسابى .

ولقد أشرنا من قبل أن الأعداد المطبوعة فى الخرج (output) ربما تحتوى على مسافات خالية قبلها أو بعدها بحسب طراز اللغة .

● **صيغة عبارة الطباعة Writeln :**

من الأمثلة السابقة نرى أن عبارة الطباعة تتبع الصيغة الآتية :

writeln(expression)

↓

بمعنى : اطبع التعبير بين القوسين

والتعبير الذى يقع بين قوسى عبارة الطباعة قد ينتمى إلى أحد الأنماط الآتية :

١ ــ حقيقى .

٢ ــ صحيح .

٣ ــ منطقى (فيما عدا بعض الطرازات) .

٤ ــ لبنة .

٥ ــ مصفوفة الحرفيات المحزمّة (packed array of char) فإذا احتوى القوس على أكثر من تعبير فإن التعبيرات تفصل عن بعضها البعض بفاصلة كما رأينا فى الأمثلة السابقة .

وإذا احتوت اللغة على المتغيرات الحرفيّة string فإنها يجوز طباعتها أيضاً .

(٣ ــ ٤) عبارة الخرج : Write :

قد نقول « عبارة الطبع » أو « عبارة الخرج » وكلاهما يؤدى نفس المعنى فكلاهما تعنى استرجاع المعلومات من ذاكرة الكومبيوتر وإخراجها على أحد أجهزة الخرج مثل الشاشة أو جهاز الطباعة .

وعبارة الطبع الجديدة write تؤدى نفس الغرض لكن فارقاً واحداً يميزها عن عبارة الطبع writeln وهو أنها لا تجعل البرنامج ينتقل إلى السطر التالى . فلو تتابعت عدة عبارات منها فإن محتويات العبارات جميعاً تطبع على نفس السطر .

مثال (٣ ــ ٥) :

النتيجة المطبوعة	العبارة
my name is Aly Hassan	write ('my name is ') ; write ('Aly Hassan')

شكل (٣ ــ ٥)

فكما نرى أن عبارتى الطبع اللتين أتيتا متتاليتين قد نتج عنهما سطر واحد فى خط الطبع .

وعلى ذلك نتوقع أن تؤدى العبارة :

writeln;

إلى ترك سطر خالٍ دون طباعة أى شيء . ومن خبر لغة بيسك من قبل لعله يلاحظ التشابه بين عبارة الطبع **PRINT** فى لغة بيسك والعبارة **writeln** فى لغة باسكال .

وينطبق على العبارة write ما ينطبق على العبارة writeln من قوانين .

مثال (٣ ــ ٦) :

النتيجة المطبوعة	العبارة
1st line 2nd line	write ('1st line'); writeln; writeln('2nd line')

شكل (٣ ــ ٦)

(٣ ــ ٥) استخدام عبارة الطبع فى الرسم :

يمكن باستخدام عبارة الطبع write أو writeln رسم ما نشاء من الأشكال على الشاشة باستخدام اللبنات المختلفة التى تتيحها اللغة .

فبين علامتى الاقتباس يمكنك أن تضع ما تشاء من اللبنات مع قيد واحد على استخدام علامة الاقتباس نفسها حيث يتطلب الأمر استخدام علامتين متتابعتين كما عرضنا فى مستهل الباب الأول .

والبرنامج التالى يرسم مثلثا باستخدام عبارة الطباعة :

مثال (٣ ــ ٧) :

البرنامج

```
program triangle(output);
begin
   writeln('                *');
   writeln('              *   *');
   writeln('            *       *');
   writeln('          *           *');
   writeln('        *               *');
   writeln('      *     triangle      *');
   writeln('    *                       *');
   writeln('  *                           *');
   writeln('* * * * * * * * * * * * * * * * *')
end.
```

شكل (٣ ـــ ٧) ا

التنفيذ

```
                *
              *   *
            *       *
          *           *
        *               *
      *     triangle      *
    *                       *
  *                           *
* * * * * * * * * * * * * * * * *
```

شكل (٣ ـــ ٧) ب

هل لاحظت فى البرنامج السابق ما جاء فى السطر الأول مع اسم البرنامج triangle ؟

لقد عقب اسم البرنامج العبارة (output) منفردة لأن هذا البرنامج لا يحتاج

لأية مدخلات (inputs) فهو مخصص للطباعة فقط !

مثال (٣ — ٨) طباعة فاتورة الكهرباء :

فلنفرض أن فاتورة الكهرباء قد جاءتك بهذا المنظر المبهج شكل (٣ — ٨) .

```
******************************
present meter reading     6015
previous meter reading    5899
units used                 116
rate per unit             3.4p
standing charge          £1.14

the sum due is           £5.08
******************************
```

شكل (٣ — ٨)

ترى ما هو البرنامج الذى فى خلفية هذه الفاتورة ؟

هناك بعض البيانات يحتاجها البرنامج بالضرورة مثل القراءة الحالية (present meter reading) والقراءة السابقة (previous meter reading)

وهناك بيانات تحتاج لإجراء بعض المعالجات الحسابية مثل المبلغ المطلوب (the sum due) والكهرباء المسحوبة (units used) وهناك بعض البيانات الثابتة التى تستخدم مع كل الفواتير مثل سعر الكهرباء (rate per unit) والمصروفات الثابتة (standing charge) .

لذلك فالبرنامج الذى نحن بصدده يحتاج لقراءة بعض البيانات من الخارج وهى القراءة السابقة والقراءة الحالية ثم يقوم بإجراء بعض الحسابات ثم يطبع الناتج بالصورة المطلوبة .

وهذا هو البرنامج :

البرنامج

فاتورة الكهرباء

```
program electricitybill(input,output);

var present, previous : integer;

begin

   read(present, previous);

   writeln('******************************');

   writeln('present meter reading    ',  present);
   writeln('previous meter reading   ', previous);

   writeln('units used               ',
                               present - previous);
   writeln('rate per unit            3.4p');
   writeln('standing charge         £1.14');

   writeln;
   write  ('the sum due is          £');
   writeln( (present - previous)*3.4/100 + 1.14 );

   writeln('******************************')
end.
```

شكل (٣ ــ ٩)

والملاحظة التى نخرج بها من هذا البرنامج هى كيفية استخدام المسافات الخالية لتنسيق الأعمدة والخانات فى الفاتورة النهائية المطلوب طباعتها .

(٣ ــ ٦) صياغة الخرج Formatted output :

هذه هى طريقة بديلة للحصول على الخرج مطبوعاً وفقاً للصورة التى نراها ، وتسمى بالطباعة المصاغة أو الخرج المصاغ (formatted) .

ولنر العبارة الآتية :

writeln(2*256:8)

الصيغة

إن الرقم 8: الذى جاء فى مؤخرة عبارة الطبع يخبر الكومبيوتر بأن يخصص الطباعة الناتج ثمانى خانات . وعلى ذلك يقوم الكومبيوتر بإجراء العملية الحسابية 2*256 حيث يكون الناتج هو **512** أى يحتل ثلاث خانات ، فيترك الكومبيوتر خمس مسافات خالية ثم يطبع العدد 512 . وهذه الطريقة تعتبر طريقة فعالة لتنظيم الطباعة لا سيما إذا كانت السطور المتتالية تنتظم فى أعمدة . وإذا أردنا كتابة البرنامج الذى استخدمناها من قبل فى رسم مثلث النجوم ـــ بهذه الطريقة فإنه يصبح كالتالى . كما فى شكل (٣ ـــ ١٠) :

مثال (٣ ـــ ٩) :

مثلث من النجوم

```
program triangle(output);
begin
     writeln('*':17);
     writeln('*':15,'*':4);
     writeln('*':13,'*':8);
     writeln('*':11,'*':12);
     writeln('*':9,'*':16);
     writeln('*':7,'triangle':13,'*':7);
     writeln('*':5,'*':24);
     writeln('*':3,'*':28);
     writeln('* * * * * * * * * * * * * * * *')
end.
```

شكل (٣ ـــ ١٠)

● صياغة الطباعة مع الأعداد الحقيقية :

أما مع الأرقام الحقيقية فالصيغة تحتوى على جزئين الجزء الأول يمثل عدد خانات العدد كلها (عدد اللبنات المطلوب طبعها) أما الثانى فيمثل عدد الأرقام العشرية التى تقع على يمين العلامة .

والمثال التالى الموضح بالبرنامج شكل (٣ ــ ١١) يوضح كيفية صياغة الأعداد الحقيقية . فقد خصصنا عددين للمتغيرين x, y ثم طبعناهما وفقاً للصيغة الواردة فى عبارة الطبع writeln. وللتأكد من النتيجة التى نحصل عليها فقد طبعنا سطراً من النجوم فوق الأعداد حتى تظهر خانة كل رقم بوضوح .

فمثلاً العدد الأول **9.4790** يحتوى على ٦ خانات بما فيها خانة العلامة العشرية لكنه ظهر فى الخرج ممثلاً فى ٨ خانات أى مسبوقاً بخانتين خاليتين (أنظر تنفيذ البرنامج) وذلك لأن الصيغة التى تم طبعه بها تحتوى على الرقم 8 يليه الرقم 4 . والرقم 8 يعنى عدد خانات الأرقام الصحيحة بما فيها العلامة ، والرقم 4 يعنى عدد الخانات العشرية كذلك نجد أن الفاصل بين العدد الأول والثانى نتج من أن العدد الثانى 3.14159 ظهر مطبوعاً فى عشر خانات وفقاً للصيغة المحددة له (10:5:) ولذلك نجد أن هذا الفاصل يحتوى على ثلاث خانات أو ثلاث نجوم وهو الفرق بين عدد الأرقام الفعلية للعدد وبين رقم الصيغة (10:) .

مثال (٣ ــ ١٠) :

البرنامج

```
var x,y,z:real;
begin
x:=9.479;y:=3.141592;
writeln('*************************************');
writeln(x:8:4,y:10:5,x+y:9:2)
end.
```

التنفيذ

```
*************************************
  9.4790    3.14159     12.62

>
```

شكل (٣ ــ ١١)

مثال (٣ ــ ١١) :

الآن يمكننا تمثيل برنامج فاتورة الكهرباء باستخدام الطباعة المصاغة كالآتى :

```
writeln('present meter reading ', present:7);
writeln('previous meter reading', previous:7);
writeln('units used             ', present-previous:7);
writeln('rate per unit          ', 3.4 :6:1, 'p');
writeln('standing charge       £', 1.14:7:2);
write  ('the sum due is        £');
writeln( (present-previous)*3.4/100 + 1.14 :7:2 )
```

شكل (٣ ــ ١٢)

■ تمرينات على الباب الثالث :

س (٣ ــ ١) : فى مثال فاتورة الكهرباء الذى عرضناه قد يكون من المناسب أن نستخدم الثوابت المسمّاه (named constants) بدلاً من الأرقام حتى يكون البرنامج أكثر عمومية ويسهل تغييره فى المستقبل إذا طرأ أى تغيير على الأرقام اكتب البرنامج مرة أخرى باستخدام الثوابت المسماه والطباعة المصاغة .

س (٣ ــ ٢) اكتب برنامجاً يقرأ درجات تلميذ فى أربعة امتحانات مختلفة (أرقاماً صحيحة) ويطبع متوسط الدرجات .

س (٣ ــ ٣) اكتب برنامجاً يقرأ عددين صحيحين ويطبع مجموعهما وحاصل ضربهما فى صورة معادلتين كالمثال الآتى :

إذا كان العددان هما :

4 7

فإن الخرج يكون على الصورة :

4+7=11
4*7=28

س (٣ ــ ٤) : اكتب برنامجاً يقرأ عدد ساعات العمل لموظف ما ، والأجر فى الساعة ، وعدد الساعات الإضافية . علما بأن أجر الوقت الإضافى ١,٥ مرة قدر معدل الأجر العادى ، ثم اطبع المرتب الإجمالى .

س (٣ ــ ٥) : اكتب برنامجاً يكتب الحرف الأول من اسمك باستخدام النجوم (*) .

س (٣ ــ ٦) : اكتب برنامجاً يستقبل رصيدك فى البنك والمبلغ المسحوب ويطبع الرصيد النهائى بعد عملية السحب .

س (٣ ــ ٧) : اكتب برنامجاً يقرأ ثمن السلعة الإجمالى ونسبة الخصم (نسبة مئوية) ويقوم البرنامج بطبع فاتورة البيع كالمثال الآتى :

```
**********************
gross price       $56.25
discount rate       2.5%
discount          $ 1.41
discount price    $54.84
**********************
```

شكل (٣ ــ ١٣)

س (٣ ــ ٨) : وفقاً للنظرية النسبية فإن الأجسام المتحركة بسرعة كبيرة قريبة من سرعة الضوء ينكمش طولها فى اتجاه سرعتها . وهذا يُعبّر عنه بالمعادلة الآتية :

$$l = lo \sqrt{1 - \frac{V^2}{C^2}}$$

حيث : l   هو طول الجسم أثناء الحركة

lo   هو طول الجسم ساكناً

V   سرعة الجسم مقاسة **بالمتر / ثانية .**

C   سرعة الضوء وهى تساوى :

299 792 458 متر / ثانية

اكتب برنامجاً بلغة باسكال للإعلان عن المتغيرات المتضمنة بهذه العلاقة .

وبفرض معلومية طول الجسم ساكناً ، وسرعته اكتب عبارة (أو عبارات) التخصيص اللازمة لحساب طول الجسم أثناء الحركة .



الباب الزابع

التفريع والتكرار

Branching & Looping

مفتتح

إن كل ما صادفناه حتى الآن من برامج عبارة عن بعض العمليات الحسابية والقرارات المنطقية البسيطة التى يستطيع الإنسان أداءها بدون احتياج حقيقى للكومبيوتر . لكننا لو اعتبرنا مشكلة واقعية مثل حساب الأجور لألف موظف يعملون فى شركة كبيرة لوجدناها مشكلة حقيقية تتطلب جهداً هائلاً ...

وربما كان حساب الأجر لموظف واحد أمر بسيط لا يستغرق دقائق ولكن عندما يزيد العدد إلى هذا الحد فسوف تواجهنا مشكلتان .. الأولى هى السرعة والثانية هى العمل المتكرر الذى يبعث على الملل وربما يسفر عن أخطاء تحتاج لمراجعات كثيرة .

وهذا هو دور الكومبيوتر الحقيقى .. فهو لا يمل ولا يتعب من التكرار ، ولا يخطىء طالما برمجته بالمنطق السليم ، هذا فضلاً عن سرعته المذهلة .

وفى هذا الباب نتحدث عن الوسائل المختلفة التى نستفيد بها من هذه الخصائص التى يتمتع بها الكومبيوتر .

while for repeat

(٤ ـــ ١) عبارات التحكم Control statements :

رأينا فى الأبواب السابقة أن بعض عبارات لغة باسكال تستخدم فى إدخال البيانات والبعض الآخر فى إخراج المعلومات وطباعتها ، كما خبرنا عبارات التخصيص والتعبيرات بأنواعها التى تعتبر الأداة الرئيسية فى معالجة البيانات .

أما العبارات التى نحن بصدد عرضها الآن فهى تهدف إلى حسن استخدام إمكانات الكومبيوتر بالاستفادة من سرعته فى إجراء العمليات ، وهى تسمى عبارات التحكم . وأولى عمليات التحكم هى عملية التكرار .. وما أكثر ما يحتاج الكومبيوتر إلى التكرار فهذا هو أحد الأهداف الرئيسية من استخدامه .

ومن عبارات التحكم أيضاً تغيير مسار البرنامج وفقاً لتحقق شرط ما . وقد نطلق على هذه العملية تفريع البرنامج .

(٤ ـــ ٢) الحلقات التكرارية For-statement :

لو أردنا أن يطبع الكومبيوتر رسالة ما عدة مرات متتالية فهناك حل سهل بأن نكتب عبارة الطبع بعدد مرات التكرار المطلوبة . ولكن هذا الحل قد يؤدى بنا إلى كتابة برنامج طويل جداً بلا داع .

الطريقة البديلة هى استخدام العبارة for كما فى المثال شكل (٤ ـــ ١) حيث يقوم البرنامج بطباعة الرسالة خمس مرات متتالية .

مثال (٤ ـــ ١) :

البرنامج :

```
program demo(output);
     var i:integer;
begin
     for i:=1 to 5 do
     begin
     writeln('SORRY..I DON''T HEAR YOU ...SAY AGAIN');
     writeln
     end
end.
```

متغير العداد

بداية الحلقة

نهاية الحلقة

التنفيذ :

```
SORRY..I DON'T HEAR YOU ...SAY AGAIN
SORRY..I DON'T HEAR YOU ...SAY AGAIN
SORRY..I DON'T HEAR YOU ...SAY AGAIN
SORRY..I DON'T HEAR YOU ...SAY AGAIN
SORRY..I DON'T HEAR YOU ...SAY AGAIN
```

شكل (٤ ـــ ١)

كيف تعمل عبارة التكرار for ؟

نرى فى السطر الأول (بعد بدء البرنامج بالعبارة begin) عبارة التكرار بالصورة الآتية :

for i:=1 to 5 do

أما المتغيّر i الذى أعلناه فى البداية كمتغير صحيح فهو يعمل كعدّاد لعدد مرات التكرار . وقد بدأ هنا بالقيمة (i=1) وسوف يوقف التكرار عندما تصل قراءته 5 .

وتشتمل العبارة for على كلمات ثلاث هى for ، to ، do ، ويلى العبارة for الحلقة التكرارية التى تحتوى على العملية المطلوب تكرارها وهى أمر الطباعة فى حالتنا هذه . ونلاحظ أن الحلقة التكرارية لها بداية begin ونهاية end بصرف النظر عن بداية ونهاية البرنامج الأصلى الذى تنتمى إليه .

وهناك عدة ملاحظات على قواعد اللغة فى هذا البرنامج :

(١) لا تستخدم الفاصلة المنقوطة (semicolon) بعد كلمة do لأن عبارة for لا تكتمل إلاّ عندما تصل إلى نهايتها (end) . أما ما يقع بينهما فيعتبر جزءاً واحداً مستقلاً عن عبارات البرنامج .

ولذلك يمكن القول بأن البرنامج يتعامل مع الحلقة التكرارية التى تقع ما بين begin ، end كأنها عبارة واحدة أو بمعنى آخر فهذا البرنامج يمكن اعتباره محتوياً على عبارة واحدة لذلك لم نستخدم الفاصلة المنقوطة بعد العبارة end التى تمثل نهاية هذه العبارة .

ولو كان البرنامج يحتوى على عبارة أخرى بعد نهاية الحلقة لأصبح من الضرورى استخدام الفاصلة بعد end لفصل العبارتين .

(٢) ما بداخل الحلقة التكرارية يعبتر برنامجاً قائماً بذاته له بداية ونهاية وتتابع فيه العبارات مفصولة بالفاصلة المنقوطة . ويسرى على الحلقة ما يسرى على البرنامج من قواعد .

(٣) مما يجدر بالملاحظة أن الإفراط فى استخدام الفاصلة المنقوطة لا يؤثر على سلامة البرنامج من الناحية اللغوية ولكن عدم وجود الفاصلة متى كان وجودها ضرورياً يمنع تنفيذ البرنامج تماماً .

(٤) ماذا يحدث لو حذفنا بداية (begin) ونهاية (end) الحلقة التكرارية ؟ .

إن فائدة البداية والنهاية كفائدة الأقواس فى التعبيرات الحسابية أى أنها تحدد العبارات المقصودة بالتكرار . فإذا حذفنا البداية والنهاية اقتصر التكرار على أول عبارة تالية للكلمة do .

(٥) يبدأ العدّاد من **قيمة ابتدائية** (1) وينتهى عند **قيمة نهائية** (5) وبتغيير القيمة النهائية يمكن تغيير عدد مرات التكرار وهذا لا يتطلب سوى إصلاح بسيط فى البرنامج .

بل إن هذه القيمة النهائية يمكن أن تكون متغيّراً يقوم البرنامج بقراءته بالعبارة read وبذلك نستغنى تماماً عن إصلاح البرنامج وإعادة ترجمته .

مثال (٤ ـــ ٢) :

```
var f,i:integer;
begin
   read (f);   <---- قراءة نهاية العدّاد
   for i:=1 to f do
      begin
      writeln('sorry..say again');
      writeln
   end
```

شكل (٤ ـــ ٢)

ملاحظة : عندما يعمل البرنامج السابق سوف يستقبل قيمة نهاية العدّاد f ويبدأ فى طباعة العبارة المطلوب تكرارها على نفس السطر . وفى هذه الحالة يفضل استخدام عبارة القراءة readln بدلاً من read حيث أنها تسمح بالانتقال إلى سطر جديد بعد القراءة . وهى تكافىء استخدام العبارة writeln مع العبارة read بهدف الانتقال إلى سطر جديد بعد القراءة .

وهذا هو البرنامج والتنفيذ :

مثال (٤ ــ ٣) :

البرنامج

```
var f,i:integer;
begin
   readln (f);        ← القراءة على سطر مستقل
   for i:=1 to f do
       begin
       writeln('sorry..say again');
       writeln
   end

end.
```

التنفيذ

```
Running
3  ←———— البيان المدخل
sorry..say again   ┐
                   │
sorry..say again   ├ ← النتيجة
                   │
sorry..say again   ┘

/

Running
5  ←———— البيان المدخل
sorry..say again   ┐
                   │
sorry..say again   │
                   │
sorry..say again   ├ ← النتيجة
                   │
sorry..say again   │
                   │
sorry..say again   ┘
```

شكل (٤ ــ ٣)

(٤ ــ ٣) خصائص الحلقة التكرارية : for

● يجوز استخدام متغيّر العدّاد لأداء وظيفة أخرى بداخل الحلقة التكرارية مثل طباعة الأرقام المسلسلة :

```
for x:=1 to 5 do write(x:4)
```

بموجب هذه الحلقة يقوم البرنامج بتكرار العبارة write(x:4) خمس مرّات وفى كل مرة يطبع قيمة متغيّر العدّاد x فى أربعة خانات . فيكون ناتج هذا البرنامج عبارة عن الأرقام من 1 إلى 5 متجاورة أى :

```
1   2   3   4   5
```

● يتزايد متغير العدّاد بخطوة مقدارها 1 عند كل تكرار لكنه قد يبدأ من أى قيمة ابتدائية سالبة أو موجبة :

```
for x:= -5 to 5 do write(x:3)
```

تقوم هذه الحلقة بطبع الأرقام من (5−) إلى (5+) متجاورة فى خانات ثلاثية كالآتى :

```
-5 -4 -3 -2 -1 0 1 2 3 4 5
```

● يتزايد عدّاد الحلقة التكرارية بخطوة مقدارها 1 ، لذلك فإذا أردنا طباعة الأعداد الزوجية تطلب ذلك شيئاً من المهارة .

```
for x:=1 to 50 do writeln(n*2:3)
```

يطبع هذا البرنامج الأعداد الزوجية متتالية تحت بعضها البعض من 2 إلى 100 .

```
for x:=1 to 50 do writeln(n*2-1:3)
```

أما هذا البرنامج فيطبع الأعداد الفردية من 1 إلى 99 . وكذلك يمكن كتابته بالطريقة الآتية :

```
for x:=0 to 49 do writeln(n*2+1:3)
```

(٤ ـ ٤) الحلقة التكرارية بخطوة سالبة :

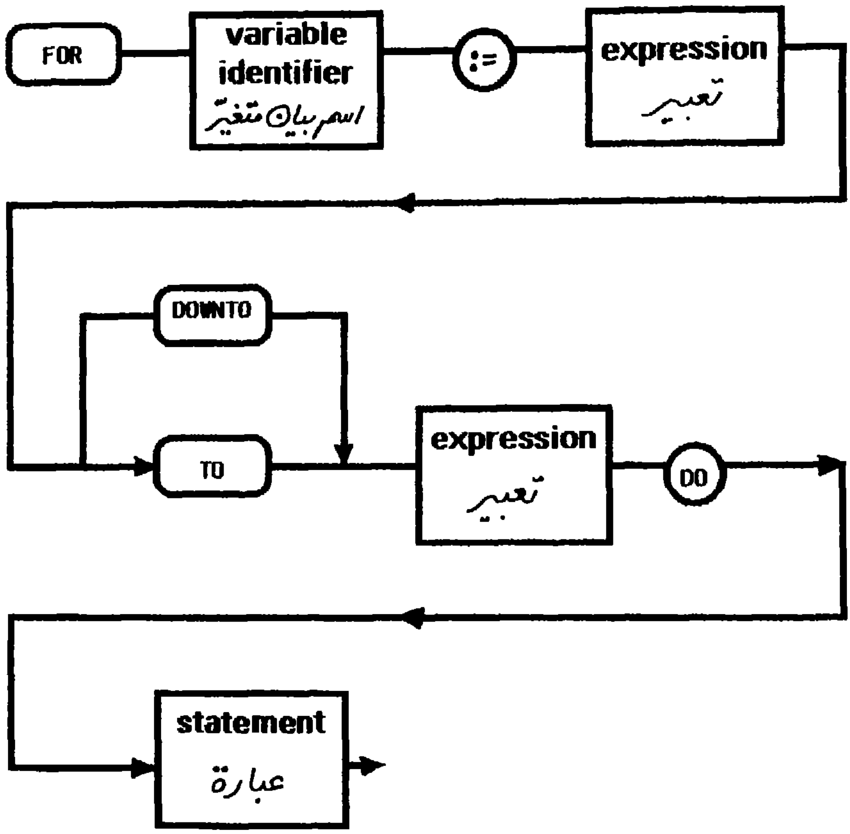
لاحظنا أن الحلقة التكرارية التى تعاملنا معها التى كان العداد يتزايد فيها تزايداً مطرداً . لذلك يقال لها أنها حلقة ذات خطوة موجبة .

ويجوز أن تكون الخطوة سالبة أيضاً بمعنى أن تتناقص قيمة متغير العدّاد فى كل مرة كالمثال الآتى :

```
for x:=100 downto 0 writeln(x:3)
```

فهنا تتناقص خطوة العدّاد من 100 حتى تصل إلى الصفر بموجب الكلمة **downto** التى حلت محل كلمة **to** فى الصيغة السابقة للعبارة for .

ويمكن التعبير عن الحلقة التكرارية بنوعيها (ذات الخطوة الموجبة) وذات الخطوة السالبة بالرسم كما فى شكل (٤ ـ ٤) .


FOR
variable identifier
اسم بيان متغير
:=
expression
تعبير
DOWNTO
TO
expression
تعبير
DO
statement
عبارة

شكل (٤ ــ ٤)

مثال (٤ ــ ٥) عمولة مندوب مبيعات :

يحصل مندوب مبيعات إحدى الشركات على عمولة بمعدل ١٢,٥٪ من سعر المبيعات التى لا يتعدى ثمنها ١٠٠ جنيه ، ويزيد هذا المعدل إلى ١٥,٥٪ على المبيعات التى يتعدى ثمنها المائة .

والبرنامج التالى يقوم بطبع العمولة التى يحصل عليها من أى عملية وحتى ٢٠٠ جنيه كحد أقصى لسعر البيع (وبالطبع يمكن تغيير هذا الحد الأقصى بحسب نوع السلعة المباعة) .

البرنامج :

عمولة مندوب مبيعات

```
program commissiontable(output);

const lowerrate = 0.125;  higherrate = 0.155;
      topoflowerrange=100;

var sale : integer;

begin
   writeln('sale   commission');  writeln;

   for sale := 1 to topoflowerrange do
      writeln(sale:4, sale*lowerrate:13:2);

   for sale := topoflowerrange+1 to 200 do
      writeln(sale:4, sale*higherrate:13:2)

end.
```

التنفيذ :

```
sale   commission

  1        0.12
  2        0.25
  3        0.37
  4        0.50
  5        0.62
  .           .
  .           .
  .           .
  .           .
 99       12.38
100       12.50
101       15.65
102       15.81
  .           .
  .           .
  .           .
  .           .
199       30.84
200       31.00
```

شكل (٤ ــ ٥)

(٤ ــ ٥) الاختيار بين البدائل (If-statement) :

من « خصائص الكومبيوتر » أنه يستطيع اتخاذ القرارات بمعنى الاختيار بين البدائل المتعددة والخروج بالقرار النهائى . وهذه الخاصية هى إمكانية من إمكانات اللغة التى نستخدمها لبرمجة الكومبيوتر .

ونحن فى غنى عن القول بأن علينا أن نخبر الكومبيوتر بكيفية اتخاذ القرار وطريقة المقارنة بين البدائل إذا أردناه أن يتخذ لنا قرار ما .

والعبارة التى تمدنا بها لغة باسكال لهذا الغرض هى العبارة if ولنر هذا المثال :

```
if age < 18 then

        write('age is less than required')
```

إن العبارة if فى هذا المثال تفصح عن معناها فهى تقول ببساطة : « إذا كان العمر أقل من ١٨ عاماً فاطبع الرسالة : السن أقل من المطلوب » والعمر أو السن يعبر عنه بالمتغير العددى age .

ولنر هذا المثال :

```
if c > 3 then

        fine := fine*2
```

وهذا قرار تختص به الشرطة حيث أنه يبحث الحالة عندما تزيد عدد مرات ارتكاب المخالفة عن ثلاثة مرات . فهنا نمثل عدد المرات بالمتغير c، فإذا زادت قيمة المتغير c عن ثلاثة قام البرنامج بمضاعفة قيمة المخالفة (fine) بموجب العبارة :

```
fine := fine*2
```

فإذا كانت قيمة المتغير fine هى ٥٤ مثلاً فإنه بموجب هذه العبارة يتم ضرب هذه القيمة فى الرقم ٢ وتخصص القيمة الناتجة لنفس المتغير fine .

وهذا مثال آخر نتعامل فيه مع أرقام مجردة :

```
if t > 100 then
   t :=t-0.10*t
```

فى هذا المثال أيضاً يتم تغيير القيمة المخزنة فى المتغير t إذا زادت عن المائة . والتغيير المطلوب هو طرح العُشر من قيمة المتغير t .

لعلنا لاحظنا فى كل الأمثلة السابقة أن العبارة if تتبع صيغة معينة هى :



شكل (٤ ــ ٦)

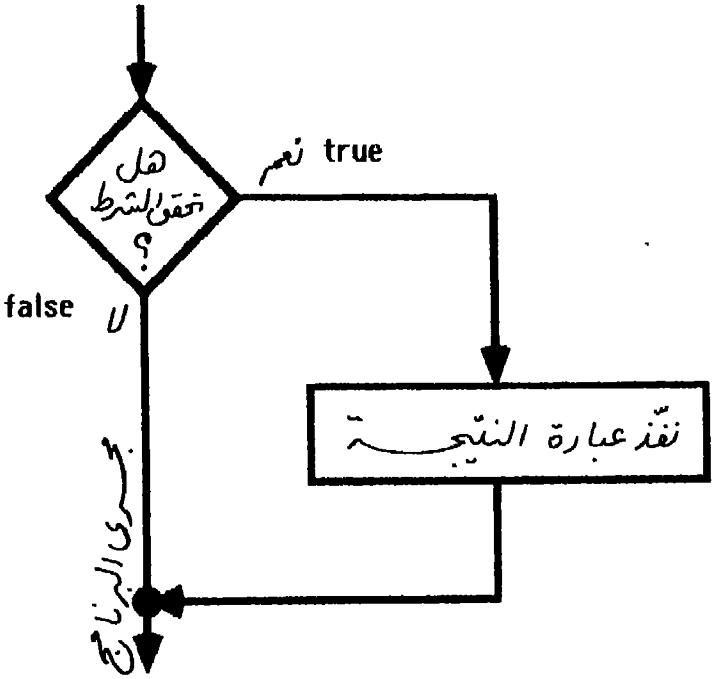
أى أنها تتكون من جزئين الأول هو الشرط والثانى هو النتيجة ، والشرط عادة يكون تعبيراً منطقياً مثل :

age<18
t>100
c>3

أما النتيجة فهى أى عبارة من عبارات اللغة التى من شأنها أن تعبر عن النتيجة المطلوبة .

ولعل بعضنا الآن يتساءل : « ماذا يفعل البرنامج إذا لم يتحقق الشرط الوارد فى البرنامج » ؟ وهذا سؤال هام جداً . فالعبارة الشرطية تعترض مجرى البرنامج لاختبار الشرط المقصود فإذا تحقق الشرط تحوّل مجرى البرنامج إلى مجرى جانبى لتنفيذ العمليات الواردة فى عبارة النتيجة . أما إذا لم يتحقق الشرط فإن البرنامج يستمر فى تسلسله العادى دون الالتفات لعبارة النتيجة .

أى أنه يمكن التعبير عن منطق العبارة if بالرسم التالى :



شكل (٤ ــ ٧)

ولكن العبارة الشرطية لها بقية !

ففى بعض المواقف قد نحتاج للتعامل مع كل من النتيجة الموجبة (true) والنتيجة السالبة (false) كل بطريقة مختلفة قبل أن ننضم إلى مجرى البرنامج .

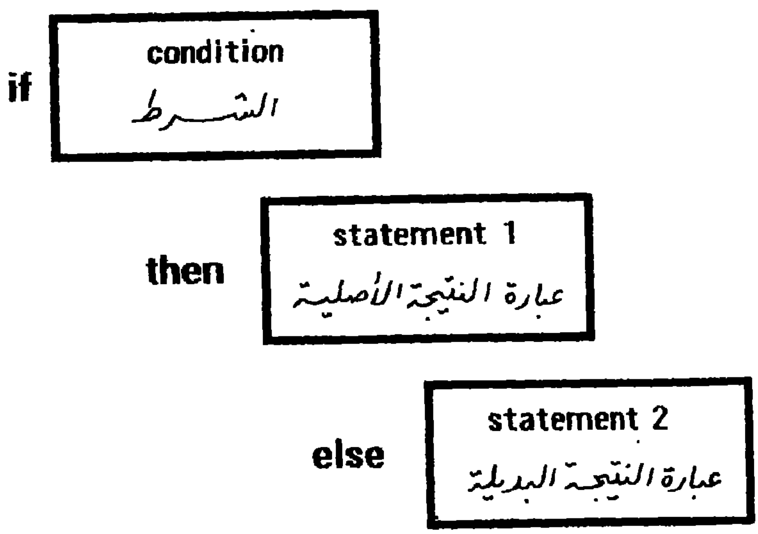
فمثلاً : قد نحتاج أن نضيف إلى المثال الأول إضافة ما مثل: «إذا كان العمر أقل من ١٨ فاطبع الرسالة :

(السن أقل من المطلوب) وإلاّ فاطبع الرسالة :
(مستوف لشرط السن) .

يمكن التعبير عن هذا الشرط المركب بالعبارة الشرطية الكاملة كالآتى :

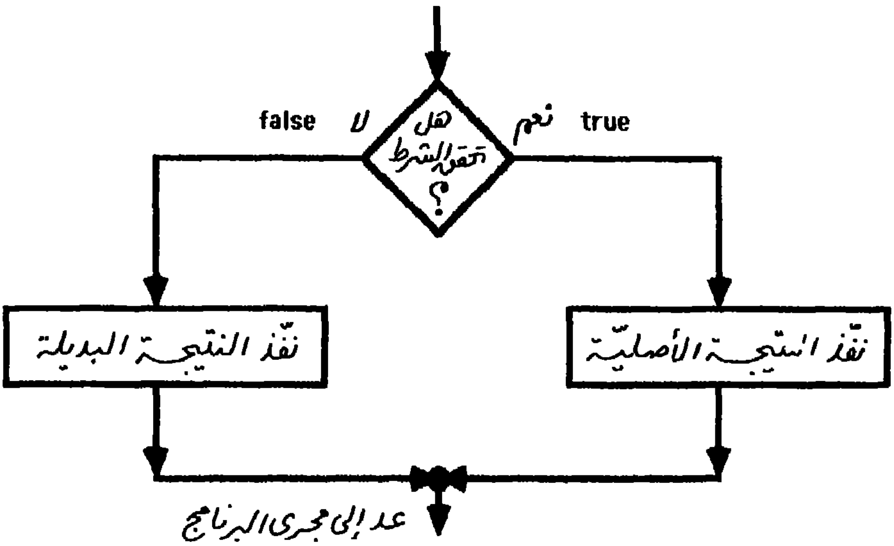
```
if age < 18 then
        write('age is less than required')
else
            write('age satisfied')
```

لقد ظهرت هنا كلمة جديدة فى العبارة الشرطية هى else بمعنى « وألاّ » والجزء الذى يتبع else يسمى عبارة النتيجة البديلة (وتسمى النتيجة الأخرى بالنتيجة الأصلية) أى أن صيغة العبارة الشرطية الكاملة كالآتى :



شكل (٤ ـ ٨)

وما يحدث فى البرنامج بالفعل أن المجرى الأصلى للبرنامج يتفرع إلى فرعين يختص أحدهما بمعالجة النتيجة الأصلية والآخر بمعالجة النتيجة البديلة ثم يجتمعان مرة أخرى فى مجرى . البرنامج الأصلى كما فى الشكل (٤ ـــ ٩) .



شكل (٤ ـــ ٩)

ولا بأس أن يحتوى البرنامج على أكثر من عبارة شرطية تأتى متتالية وراء بعضها البعض وهى فى هذه الحالة تشبه سلسلة الاختبارات التى يمر بها الطالب المتقدم للكليات العسكرية !

(٤ ـــ ٦) تطبيقات على العبارة الشرطية :

مثال (٤ ـــ ٦) مقارنة ثلاثة أعداد :

يقوم هذا البرنامج بقراءة ثلاثة أرقام وطباعة أكبر رقم . ومتروك لك تتبع منطق البرنامج علماً بأن الأرقام الثلاثة تستقبل فى المتغيرات المعبرة عن أسمائها :

first, second, third

كما تم استخدام المتغير largest كوعاء لكى نضع فيه الرقم الأكبر ، والمتغير largestsofar كوعاء مؤقت أثناء المقارنات وهذا الاسم يعنى « الأكبر حتى الآن ... » .

```
program largest(input,output);

var first, second, third, largestsofar, largest : real

begin
   read(first, second, third);

   if first > second then
      largestsofar := first
   else
      largestsofar := second;

   if largestsofar > third then
      largest := largestsofar
   else
      largest := third;

   writeln('biggest number is ', largest)
end.
```

مقارنة ثلاثة أعداد

شكل (٤ ـــ ٦)

قبل أن ننتقل إلى المثال الثانى دعنا نتحدث قليلاً عن الفصلات المنقوطة

(semicolons) وهى أمر قد يشغل بال من كان جديداً على التعامل مع لغة باسكال .

لعلنا لاحظنا أن الفاصلة المنقوطة لم تظهر خلال العبارة الشرطية إلاّ عند نهايتها أى بعد النتيجة البديلة . ذلك لأن العبارة الشرطية سواء كانت **بسيطة أو كاملة** تعتبر عبارة واحدة متصلة رغم احتوائها على عبارات كاملة بداخلها .

وهذه الملاحظة من الأهمية بمكان . فقد ذكرنا من قبل أنه لا بأس من أن نفرط فى استخدام الفاصلة المنقوطة حيث لا يتطلب الموقف ذلك مثل استخدامها مع العبارة الأخيرة فى البرنامج . ولكن هذا لا يعنى بحال أن تأتى الفاصلة المنقوطة فى منتصف عبارة ما ، فهذا يفسد معنى العبارة . **لذلك لا يجوز أن يتخلل العبارة الشرطية بأجزائها أية فاصلة منقوطة حتى يستطيع الكومبيوتر التعامل معها ككل موّحد .**

مثال (٤ ــ ٧) المزيد من المقارنات :

يقوم هذا البرنامج باستقبال أربعة أرقام . سوف نطلق على الرقم الأول الاسم standard بمعنى الرقم القياسى وسوف يتم استقبال الأرقام الثلاثة الأخرى واحداً بعد الآخر فى وعاء واحد (متغير واحد) يحمل الاسم mext .

وبعد استقبال كل رقم من الأرقام الثلاثة يتم اختبار الفرق بينه وبين الرقم القياسى فإذا كان هذا الفرق أصغر من العُشر فإن قراءة العدّاد numberclose (وهو متغير صحيح) تزداد بمقدار 1 بموجب العبارة :

```
numberclose  :=  numberclose + 1;
```

(بمعنى أضف واحداً إلى قيمة المتغير numberclose وخصص القيمة الجديدة لنفس المتغيّر) .

وبذلك فإن هذا البرنامج يقوم بحصر عدد الأرقام التى تقترب قيمتها من قيمة الرقم القياسى standard .

فإذا كان لدينا الأرقام الأربعة الآتية :

3.156 3.051 3.152 3.091

فإن خرج البرنامج يكون كالآتى :

2 values are near the standard

البرنامج

المراقبة الرقمية للإنتاج

```
program tolerance(input,output);

var standard, next : real;  numberclose : integer;

begin
   numberclose := 0;
   read(standard);

   read(next);
   if abs(standard - next) < 0.1 then
      numberclose := numberclose + 1;

   read(next);
   if abs(standard - next) < 0.1 then
      numberclose := numberclose + 1;

   read(next);
   if abs(standard - next) < 0.1 then
      numberclose := numberclose + 1;

   writeln(numberclose,
          ' values are near the standard')
end.
```

شكل (٤ ــ ١١)

ويستخدم هذا البرنامج فى المراقبة الرقمية للإنتاج حيث تقارن أبعاد المنتجات بالأبعاد القياسية المخزنة فى ذاكرة الكومبيوتر .

ولعلنا لاحظنا فى هذا البرنامج تكرار عبارة قراءة المتغير والعبارة الشرطية .

وبالطبع هناك طريقة أفضل لأداء نفس البرنامج وذلك بوضع كل العبارات المتكررة بداخل حلقة تكرارية فيصبح البرنامج كما فى شكل (٤ ـ١٢) .

وبتغيير الحد النهائى لمتغيّر العدّاد count يمكن التحكم فى عدد مرات التكرار كيفما نشاء (سوف يتم التعرض لهذه المنشئات فى الباب الخامس بالتفصيل) .

البرنامج

المراقبة الرقمية للإنتاج ؟

```
program tolerance2(input,output);

var standard, next : real;
    numberclose, count : integer;

begin

   numberclose := 0;
   read(standard);

   for count := 1 to 3 do
   begin

      read(next);
      if abs(standard - next) < 0.1 then
         numberclose := numberclose + 1

   end;
writeln(numberclose, ' values are near the standard.')
end.
```

شكل (٤ ـ ١٢)

مثال (٤ ـ ٨) اختبار السنة الكبيسة :

يحتوى البرنامج على متغيرين الأول هو year وهو متغير صحيح يمثل السنة أما الآخر فهو leap وهو متغير منطقى لتمثيل حالتى السنة : « كبيسة أو غير كبيسة » ، فهو يأخذ القيمة الصحيحة true إذا كانت السنة كبيسة ويأخذ القيمة غير الصحيحة false إذا لم تكن كذلك .

والشرط الوارد فى البرنامج لاختبار السنة الكبيسة أنه إذا كان العدد المعبر عن السنة يقبل القسمة على 4 ولا يقبل القسمة على **100** فى نفس الوقت تكون السنة كبيسة ويأخذ المتغير leap القيمة الصحيحة true . ولكن هناك حالة أخرى تصبح فيه السنة كبيسة وهى أن تقبل القسمة على **400** . ولذلك نلاحظ أن التعبير المنطقى الذى جاء فى البرنامج تعبير مرّكب ويتكون أساساً من شرطين بينهما المؤثر المنطقى OR .

فإذا تتبعنا البرنامج سطراً بسطر نجد أنه يبدأ بالإعلانات ثم يلى ذالك جسم البرنامج الرئيسى بعد البداية Begin .

تبدأ عمليات المعالجة بقراءة السنة بالعبارة read يليها طبع سطر خال بواسطة العبارة writeln . ورغم أن هذه الخطوة لا تمت إلى موضوع البرنامج بصلة ، لكنها عادة حميدة أن تتبع العبارة read بسطر خال حتى لا تختلط الكتابة على الشاشة ولكى تستطيع تمييز المدخلات التى تُغذى بها البرنامج من المعلومات التى يطبعها الكومبيوتر على الشاشة .

السطر التالى هو تخصيص التعبير المنطقى للمتغير leap وهو يقول :

« إذا كانت السنة تقبل القسمة بدون باق على العدد 4 وكانت لا تقبل القسمة بدون باق على العدد 100 .

أو

إذا كانت السنة تقبل القسمة بدون باق على العدد 400 فإن المتغير leap يأخذ القيمة true » .

يلى ذلك العبارة الشرطية if وقد جاءت هنا بسيطة للغاية :

if leap then.....

بمعنى أنه إذا كان المتغير leap يحتوى على القيمة true .. فاطبع كذا وكذا ...

كما ظهر الجزء المكمّل للعبارة else متضمناً النتيجة البديلة عندما لا يكون

المتغير leap صحيحاً .

البرنامج

```
program leapyear(input,output);

var year:integer;
    leap:boolean;
begin
     read(year);
     writeln;
     leap:=(year mod 4=0) and (year mod 100<>0)
             or (year mod 400=0);
     if leap then writeln(year,' is a leap year')
             else writeln(year,' is not a leap year')
end.
```

اختبار السنة الكبيسة

التنفيذ لسنوات مختلفة :

الرقم المدخل

```
1988
1988 is a leap year
```

النتيجة... سنة ١٩٨٨ سنة كبيسة

```
>

2000
2000 is a leap year

>

1986
1986 is not a leap year
```

شكل (٤ ــ ١٣)

ويمكنك اختبار النواتج التى حصلنا عليها من البرامج باختبار السنوات المدخلة وهى 1988 ، 2000 ، 1986 وذلك بإجراء الحسابات على آلة حاسبة أو يدوياً لمعرفة ما إذا كانت هذه السنوات كبيسة أم لا .

وهناك طريقة مثلى لاختبار مجموعة كبيرة من الأرقام الممثلة للسنوات وذلك باستخدام الحلقات التكرارية بتعديل البرنامج تعديلاً طفيفاً ليكون كما هو موضح فى شكل (٤ — ١٤) .

(البرنامج)

```
program leapyear(input,output);

var year:integer;
    leap:boolean;
begin

 for year:=1970 to 1990 do
begin
leap:=(year mod 4=0) and (year mod 100<>0)
        or (year mod 400=0);
if leap then writeln(year,' is a leap year')
   else writeln(year,' is not a leap year')
end

end.
```

(اختبار السنة الكبيسة ؟)

شكل (٤ — ١٤)

وعند تنفيذ هذا البرنامج فإن الحلقة التكرارية التى تبدأ بالرقم 1970 وتنتهى بالرقم 1990 هى التى تغذى البرنامج بالأرقام الممثلة للسنوات ، وعلى ذلك يكون ناتج هذا البرنامج كما هو موضح فى شكل (٤ — ١٥) .

تنفيذ البرنامج لعدة سنوات

```
1970  is  not  a  leap  year
1971  is  not  a  leap  year
1972  is  a  leap  year
1973  is  not  a  leap  year
1974  is  not  a  leap  year
1975  is  not  a  leap  year
1976  is  a  leap  year
1977  is  not  a  leap  year
1978  is  not  a  leap  year
1979  is  not  a  leap  year
1980  is  a  leap  year
1981  is  not  a  leap  year
1982  is  not  a  leap  year
1983  is  not  a  leap  year
1984  is  a  leap  year
1985  is  not  a  leap  year
1986  is  not  a  leap  year
1987  is  not  a  leap  year
1988  is  a  leap  year
1989  is  not  a  leap  year
1990  is  not  a  leap  year
```

شكل (٤ ــ ١٥)

(٤ ــ ٧) العبارة الشرطية المتعددة النتائج :

استخدمنا العبارة **الشرطية البسيطة** ذات النتيجة الواحدة وكذلك العبارة **الشرطية الكاملة** ذات النتيجتين الأصلية والبديلة .

والواقع أن نتيجة العبارة الشرطية بنوعيها يمكن أن تتعدد ، بمعنى أنه يمكن أن يترتب على تحقق الشرط (أو عدم تحققه) عدة نتائج متتابعة . فى هذه الحالة

فإننا سوف نحتاج أن نفصل هذه النتائج المتتابعة عن بقية عبارات البرنامج بوضعها بين عبارتى بداية (begin) ونهاية (end) اللتين تعملان عمل الأقواس فى التعبيرات الحسابية .

والبرنامج التالى تطبيق جيد للعبارة الشرطية المتعددة النتائج وهو يستقبل عددين ويطبع مجموعهما والفرق بينهما . ويبدأ البرنامج بقراءة العددية وتخصيص أحدهما للمتغير larger (بمعنى الأكبر) والآخر للمتغير smaller (بمعنى الأصغر) .

بعد ذلك تتم مقارنة العددين للتأكد من أن الأصغر يحتل المتغير smaller والأكبر يحتل المتغير larger فإن لم يكن كذلك ، فتجرى عملية تبديل لقيم المتغيرات .

وعملية التبديل تتم باستخدام المتغير الوسيط temporary ، تماماً كما لو كان لدينا كوباً من الشاى وآخر من القهوة فإننا إذا أردنا تبديل محتويات الكوبين فلا بد لنا من الاستعانة بكوب ثالث لإجراء عملية التبديل .

وتجرى عملية التبديل كنتيجة لتحقق الشرط فى العبارة if بموجب العمليات الثلاثة الآتية :

```
temporary: = larger;
larger: = smaller;
smaller: = temporary;
```

حيث تبدأ العملية « بتفريغ » قيمة المتغير larger فى المتغير المؤقت temporary ثم تفريغ محتويات المتغير smaller فى المتغير larger ثم تفريغ محتويات المتغير المؤقت فى المتغير smaller .

بهذه الخطوات الثلاث تكون محتويات المتغيرين smaller ، larger قد تم تبديلهما .

مثال (٤ ــ ٩) :

```
program twonumbers(input,output);

var larger, smaller, temporary, sum,
       difference : integer;

begin
   read(larger, smaller);

   if larger < smaller then
   begin
      temporary := larger;
      larger    := smaller;
      smaller   := temporary
   end;

   sum        := larger + smaller;
   difference := larger - smaller;

   writeln('the sum is ', sum);
   writeln('the difference is ', difference);
   writeln('nos. in order are ', larger, smaller)
end.
```

← العبارة الشرطية متعددة النتائج

شكل (٤ ــ ١٦)

(٤ ــ ٨) الاختيار بين البدائل المتعددة case :

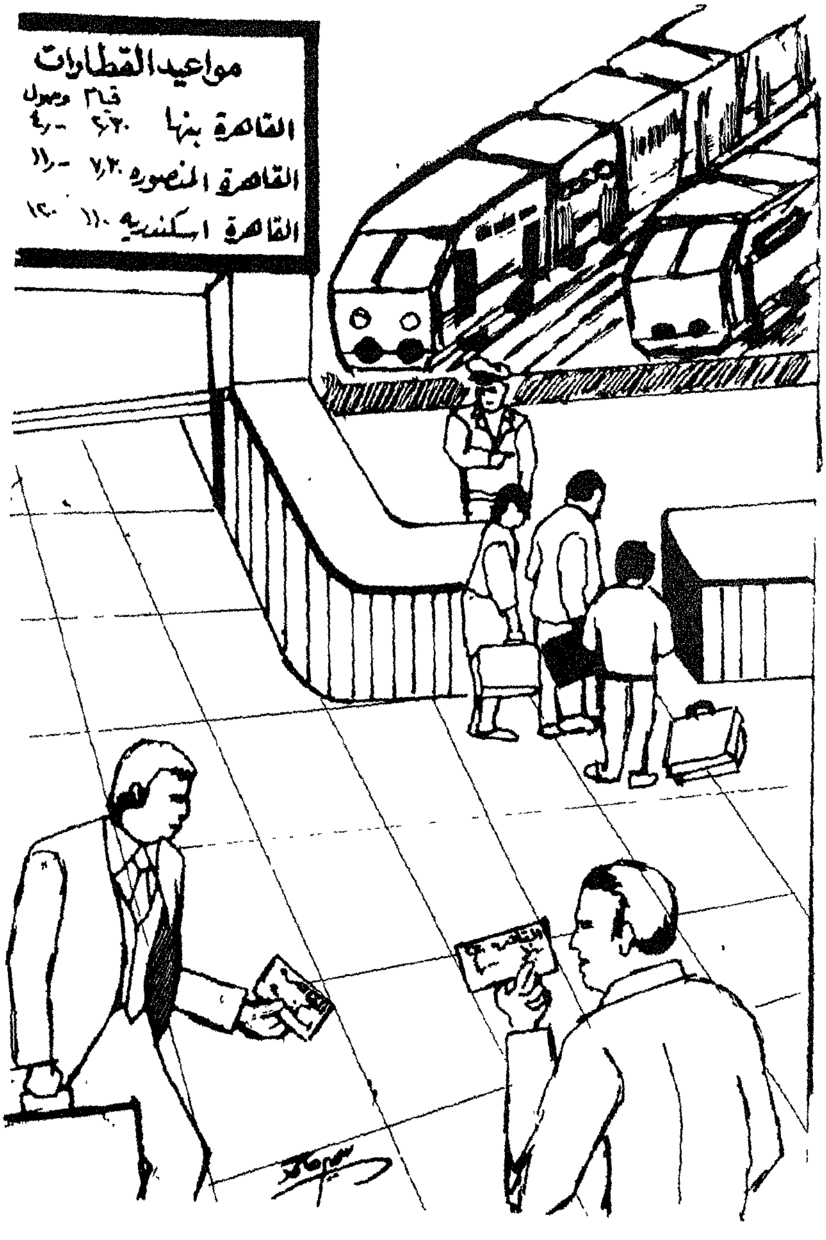
العبارة الشرطية التى تعاملنا معها حتى الآن تحتوى دائماً على بديلين فقط والاختيار بينهما يتم وفقاً لتحقق شرط واحد سواء كان بسيطاً كالمقارنة بين عددين أو كان شرطاً مركباً مثل شرط السنة الكبيسة .

هذان البديلان كأنهما وجهان لعملة واحدة وما أكثر ما نلتقى بهما فى حياتنا اليومية مثل « الخير والشر » ، « الحرب والسلام » ، « النجاح والفشل » .

فلو أنك هبطت محطة كبيرة للسكك الحديدية مثل محطة القاهرة فسوف تدهمك أفواج المسافرين الذين يتفرعون أو ينضمون فى مسالك مختلفة وعشرات القطارات المتأهبة للتحرك . وعليك فى هذا الزحام الهائل أن تتحقق من شرط معيّن حتى تصل إلى القطار المطلوب الذى يرحل بك إلى مقصدك فى سلام . إن هذا الشرط الذى تتحقق منه عادة هو رقم الرصيف وساعة القيام ، وبتحقق هذا الشرط فإنك تختار بديلاً واحداً (قطاراً واحداً) من بين عشرات البدائل الأخرى التى ربما تنطبق شروطها على مسافرين آخرين .

وأحد الوسائل التى يمكننا بها برمجة مثل هذه الحالة هى استخدام عدد من العبارات الشرطية مساو لعدد الحالات المختلفة ولكنْ يا له من عمل شاق .

ولأن هذا التطبيق هو أحد التطبيقات الهامة للكومبيوتر فإن لغة باسكال تمدنا بوسيلة فعالة فى مثل هذه الحالات وهى العبارة المركبة case . ومن سبق له دراسة لغة بيسك من قبل فإنه ولا بد قد تعرّف إلى العبارة ON GOTO التى تؤدى نفس الغرض :


مواعيد القطارات
قيام وصول
القاهرة بنها ٦،٢٠ ٨،٠٠
القاهرة المنصوره ٧،٢٠ ١١،٠٠
القاهرة اسكندرية ١١،٠ ١٢،٠

مثال (٤ ـــ ١٠) :

ولنبدأ بهذا المثال عن ماكينة المشروبات التى تغذيها بقطع العملة المختلفة فتقوم الماكينة بحساب قيمة ما دخل إليها من نقود . وبالضغط على زر ما تستطيع اختيار ما تشاء من المشروبات أو المأكولات فتقوم الماكينة فى هذه الحالة بإخراج الصنف المطلوب وكذلك إخراج باقى الحساب فى وعاء صغير مخصص للنقود .

كيف تعمل هذه الماكينة ؟

إن المبدأ الرئيسى هو مقارنة الأوزان المختلفة لقطع العملة وتحديد قيمة كل قطعة تبعاً لوزنها .

وسوف نفترض أن العملات المستخدمة هى :

العملة ذات القيمة ١٠ قروش وتزن ٣٥ جراماً .

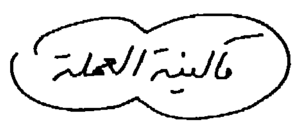
العملة ذات القيمة ٥ قروش وتزن ١٦ جراماً .

العملة ذات القيمة ١ قروش وتزن ٩ جراماً .

العملة ذات القيمة $\frac{1}{2}$ قروش وتزن ٧ جراماً .

فى هذه الحالة يمكن استخدام العبارة case فى برنامج باسكال كالآتى :

```
read (weight);
case weight of
35: amount:=amount+10;
16: amount:=amount+ 5;
9 : amount:=amount+ 1;
7 : amount:=amount+0.5;
end
```

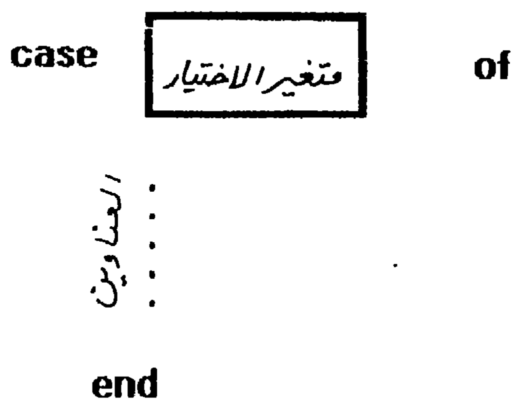


شكل (٤ ـــ ١٧)

فإذا كان وزن العملة المدخلة هو ٣٥ جراماً فإن البرنامج يضيف مقدار ١٠ قروش إلى المبلغ المخزن فى المتغير amount وإذا كان الوزن هو ١٦ جراماً يضيف البرنامج مقدار ٥ قروش وهكذا .

ويمثل المتغير amount جملة القيمة المالية المدخلة إلى الماكينة والتى على أساسها يتم الشراء واستعادة ما تبقى من نقود لدى الماكينة .

والمتغير weight يمثل وزن العملة المدخلة بالجرام . وسوف نعود إلى هذا البرنامج مرة أخرى لنستكمل بقية خطواته عندما نمضى فى عبارات باسكال أبعد من ذلك . لكن ما يهمنا فى الوقت الحالى هو الصيغة التى نكتب بها العبارة case والتى يمكن تقنينها كالآتى :



شكل (٤ ــ ١٧)

فالعبارة case تمثل برنامجاً صغيراً يقع بداخل البرنامج الرئيسى ، وهى تبدأ بكلمة case يعقبها متغير الاختيار الذى تتعدد النتائج تبعا لقيمته (وهو المتغير

weight فى المثال السابق) يليه كلمة of .

بعد ذلك تتالى العبارات التى تصف منطق الاختيار والتى تبدأ كل منها « بعنوان » رقمى يمثل قيمة معينة لمتغير الاختيار ويتبع العنوان نقطتان (:) ثم العبارة المطلوب تنفيذها إذا وقع الاختيار على العنوان الذى يبدأ به السطر .
وتنتهى العبارة المركبة case بكلمة end لتسلم دفة التنفيذ بالبرنامج الرئيسى .
ويمكن أن يضم السطر الواحد أكثر من عنوان .

مثال (٤ ـــ ١١) :

المثال التالى تستخدم فيه العبارة case لطبع إحدى ثلاث رسائل الرسالة الأولى هى :

« موسم ذو معدل منخفض » Low season rate

والثانية هى : « موسم ذو معدل متوسط » Mid season rate

والثالثة هى : « موسم ذو معدل عال » Peak season rate

وهذه الرسائل الثلاثة تناظر المواسم التجارية التى تدور مع أشهر السنة ، فالشهور يناير وفبراير ونوفمبر وديسمبر تناظر الموسم المنخفض . أما الشهور مارس وأبريل ومايو وأكتوبر فهى تناظر الموسم المتوسط ، والشهور يونيو ويوليو وأغسطس وسبتمبر فهى تمثل الموسم المرتفع المعدّل .

وهذه هى فقرة البرنامج المعبرة عن هذا المنطق :

```
read(month);

case month of
    1,2,11,12  : writeln('low season rate');
    3,4,5,10   : writeln('mid season rate');
    6,7,8,9    : writeln('peak season rate')
end
```

شكل (٤ ـــ ١٨)

أما متغير الاختيار فلا يكون بالضرورة ثابتاً عددياً بل من الجائز استخدام التعبيرات الحسابية والمنطقية أيضاً .

مثال (٤ ــ ١٢) :

فى المثال التالى نستخدم تعبيراً حسابياً للتعبير عن دخل الفرد الذى تتحدد على أساسه الضريبة التصاعدية .

(**ملاحظة** : هذا المثال لمجرد التدريب ولا يمثل أى قانون حقيقى للضرائب) ولنعتبر الحالات الآتية على سبيل المثال :

- إذا كان الدخل بين **1000** ، **0** كانت نسبة الضريبة 0 %
- إذا كان الدخل بين **2000** ، **1000** كانت نسبة الضريبة 5 %
- إذا كان الدخل بين **3000** ، **2000** كانت نسبة الضريبة 7.5 %
- إذا كان الدخل بين **4000** ، **3000** كانت نسبة الضريبة 10 %
- إذا كان الدخل بين **5000** ، **4000** كانت نسبة الضريبة 12 %

ولبرمجة هذه الحالات نستخدم متغيراً صحيحاً هو income يمثل الدخل وآخر tax يمثل نسبة الضريبة . وبالاستفادة من خاصية القطع truncation لعملية القسمة الحقيقية يمكن كتابة البرنامج التالى :

```
program tax(input,output);
var tax,income:real;
begin
    read (income);

    case trunc(income/1000) of
      0:tax:=0;
      1:tax:=5;
      2:tax:=7.5;
      3:tax:=10;
      4:tax:=12;
    end;
   writeln;
   writeln('tax to pay: ',income*tax/100)
end.
```

شكل (٤ ــ ١٩)

ونلاحظ فى هذا البرنامج أن أى عدد أقل من **1000** يعطى صفراً عند قسمته على **1000** . وإذا كان العدد أكبر من **1000** بأى نسبة فإنه يعطى الرقم 1 وهكذا ....

مثال (٤ ــ ١٣) :

أما فى هذا المثال فنستخدم فيه التعبيرات المنطقية للتعبير عن نفس مضمون البرنامج شكل (٤ ــ ١٠) .

ولكن باستخدام منطق جديد هذه المرة :

```
case first > second of
   true : largestsofar := first;
   false: largestsofar := second
end;

case largestsofar > third of
   true : largest := largestsofar;
   false: largest := third
end;

writeln('largest value is ', largest)
```

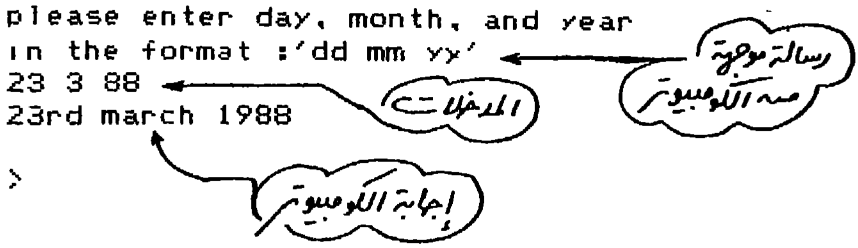
مقارنة ثلاثة أعداد

شكل (٤ ــ ٢٠)

ومن الجدير بالذكر أنه قد يتطلب البرنامج ، عند قيمة معينة لمتغير الاختيار ، تنفيذ أكثر من عبارة . وفى هذه الحالة فقد علمنا من قبل أن العبارتين begin ، end تستخدمان كما تستخدم الأقواس فى الحسابات لحصر مجموعة معينة من العبارات وعزلها عن متسلسل البرنامج الأصلى .

مثال (٤ ــ ١٤) :

لنتصور برنامجاً يعمل بالصورة الآتية كما فى الشكل التالى :



شكل (٤ ــ ٢١)

لقد بدأ البرنامج بأن سألنا إدخال تاريخ اليوم معبراً عنه بالفورمات dd mm yy' . أى أن كل من اليوم والشهر والسنة يتم التعبير عنه بعدد مكون من رقمين (على الأكثر) . لذلك قمنا بإدخال الأرقام الموضحة وهى تتكون من :

العدد 23 للتعبير عن اليوم .

العدد 3 للتعبير عن الشهر (مارس) .

العدد 88 للتعبير عن العام (١٩٨٨) .

فكانت إجابة الكومبيوتر هى :

23rd march 1988

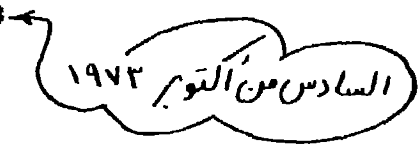
لقد استبدل الشهر رقم ٣ بالاسم « مارس » وأضاف ١٩ إلى العدد 88 ليكون 1988 كما نلاحظ أنه أضاف حرفين (rd) للعدد 23 المعبر عن اليوم ، حتى يمكن قراءة العدد على النحو التالى :

twenty third of march 1988

فلنر معاً كيف يكون الحال مع تاريخ آخر :

```
please enter day, month, and year
in the format :'dd mm yy'
6 10 73
6th october 1973

>
```



شكل (٤ ــ ٢٢)

إن هذا اليوم هو السادس من أكتوبر ١٩٧٣ . ونلاحظ أنه قد وضع الحرفين th أمام الرقم 6 هذه المرة حتى تُقرأ :

sixth of october 1973

كيف يمكن بناء هذا البرنامج ؟

إن المطلوب من البرنامج التعرف على الشهور لاستبدال كل رقم يدل على شهر باسم هذا الشهر .

كما هو مطلوب إضافة نهايات مناسبة للأيام . فمع اليوم الأول (1) أو الحادى والعشرين (21) أو الحادى والثلاثين (31) تضاف النهاية st لتدل على كلمة first . ومع الأيام (2) ، (22) تضاف النهاية nd لتدل على كلمة

. second

ومع الأيام (3) ، (23) تضاف النهاية rd لتدل على كلمة third .

أما عدا ذلك فتضاف النهاية المعتادة th .

ومع العام فالمطلوب شيء بسيط وهو طبع بداية للعدد الدال على العام وهو العدد (19) .

وهذا هو البرنامج :

البرنامج

```
program date(input,output);
var d,m,y:integer;
begin
    writeln('please enter day, month, and year');
    writeln('in the format :''dd mm yy''');
    read (d,m,y);          <---- المدخلات
    writeln;
    case d of
      1,21,31:write(d,'st');   }
      2,22   :write(d,'nd');   }  طباعة اليوم
      3,23   :write(d,'rd');   }
      4,5,6,7,8,9,10,11,12,13,14,15,16:write(d,'th');
      17,18,19,20,24,25,26,27,28,29,30:write(d,'th');
    end;
    case m of
      1:write(' january');     }
      2:write(' february');    }  طباعة الشهر
      3:write(' march');       }
      4:write(' april');       }
      5:write(' may');         }
      6:write(' june');        }
      7:write(' july');        }
      8:write(' august');      }
      9:write(' september');   }
      10:write(' october');    }
      11:write(' november');   }
      12:write(' december');   }
    end;
   writeln(' 19',y);        <---- طباعة العام
end.
```

شكل (٤ ــ ٢٣)

نلاحظ فى هذا البرنامج ما يلى :

١ ـــ إن السطر الذى يطبعه الكومبيوتر والذى يعبر عن التاريخ يُطبع على ثلاث مراحل فاليوم يطبع مع الروتين case المخصص لاختبار البدائل المختلفة للأيام .

أما الشهر فيطبع أثناء الروتين case الثانى المخصص لاختبار البدائل المختلفة للشهور .

أما العام فيطبع فى نهاية البرنامج .

ولأن عملية طباعة سطر واحد تتم على ثلاث مراحل لذلك فإننا قد استخدمنا العبارة write بدلاً من writeln عند طباعة اليوم والشهر وذلك حتى لا يتم الانتقال إلى السطر التالى .

أما عند طباعة العام فقد استخدمنا writeln حيث وصلنا إلى نهاية السطر .

٢ ـــ المتغيّرات المستخدمة هنا هى :

d : للتعبير عن اليوم
m : للتعبير عن الشهر
y : للتعبير عن العام

٣ ـــ فى الروتين case الأول نلاحظ أن الحالة التى تطبع فيها النهاية 'th' قد قسمت إلى حالتين ، وهذا لمجرد تنظيم شكل البرنامج لكنه يجوز أن يتضمن السطر الواحد أى عدد من البدائل .

٤ ـــ عند إدخال قيم هذه المتغيرات يراعى ترك مسافة خالية بين كل عدد وآخر ثم الضغط على الزر ENTER (أو RETURN) فى نهاية الإدخال .

والعدد المدخل يتكوّن من رقمين على الأكثر بمعنى أن شهر مارس مثلاً يمكن كتابته 03 ويجوز كتابته أيضاً 3 .

وهذه بعض نتائج لتنفيذ البرنامج :

تنفيذ البرنامج

```
please enter day, month, and year
in the format :'dd mm yy'
06 10 73
6th october 1973

>
```

التاريخ المدخل

اجابة الكمبيوتر

تنفيذ ١

```
please enter day, month, and year
in the format :'dd mm yy'
21 03 88
21st march 1988

>
```

تنفيذ ٢

```
please enter day, month, and year
in the format :'dd mm yy'
02 01 88
2nd january 1988

>
```

تنفيذ ٣

شكل (٤ ـ ٢٤)

(٤ ـ ٩) الحلقات التكرارية المشروطة *Conditional loops* :

عرفنا من قبل الحلقة التكرارية for التى تستخدم لتكرار تنفيذ عملية بعينها

عدداً محدوداً من المرّات .

وعرفنا العبارة الشرطية if التى بموجبها يمكن تحويل مسار البرنامج إلى مجرى معيّن وفقاً لتحقق (أو عدم تحقق) شرط معيّن .

كما عرفنا العبارة الشرطية متعددة البدائل case التى بموجبها يتفرع البرنامج إلى عدة مسالك كل وفقاً لشرط معيّن .

وفى هذه الفقرة نتعرف بعبارتين جديدتين للتكرار المشروط هما العبارتان **repeat** ، **while** . وتتميز هاتان العبارتان باحتوائهما على عمليتى الشرط والتكرار معاً .

(٤ ــ ٩ ــ ١) عبارة repeat-until :

تفيدنا هذه العبارة عندما نريد من البرنامج أن يكرر عملية معينة حتى يتحقق شرط ما ، تماماً كما نطلب من شخص ما أن يستمر فى تعبئة وعاء ما حتى يمتلىء . فعملية تعبئة الوعاء عملية **متكررة** لكنها مرهونة بتحقق **شرط** امتلاء الوعاء .

وفى اللغة العادية نقول : « **املأ هذا الوعاء حتى يمتلىء** » .

وفى لغة باسكال نقول :

repeat.......until.....

بمعنى **كرّر** هذه العملية **حتى** يتحقق شرط معيّن .

مثال (٤ ــ ١٥) قطار فى موعد الغذاء :

يريد أحد رجال الأعمال أن يستقلّ قطاراً من الاسكندرية إلى القاهرة فى حدود الساعة ١٢ ظهراً بحيث يستغل فترة تواجده بالقطار فى تناول طعام الغذاء .

ولو تصورنا أنه سوف يستعين بالكومبيوتر للبحث عن القطار المناسب بسرعة ، سواء بنفسه أو عن طريق موظف الاستعلامات فى محطة السكة الحديد ، فإننا يمكن أن نتصور شكل البرنامج الذى يؤدى العمل . إن المطلوب من البرنامج هو « موعد أول قطار بعد الثانية عشرة » ... هذا هو الشرط .

أما عملية البحث فهى تتم خلال مواعيد القطارات كلها التى عادة ما تكون مخزنة فى « ملف » (file) يقوم الكومبيوتر بالقراءة منه آلياً حتى يعثر على الموعد المطلوب .

ولو افترضنا أن جدول المواعيد يحتوى على الأرقام التالية :

وصــول	قيــام
1100	0806
1203	0901
1502	1155
1527	1215
1755	1450
1950	1645

فإنه بالبحث فى هذا الجدول نجد أن القطار المناسب هو القطار الذى يقوم الساعة 1215 (الثانية عشرة والربع) ويصل الساعة 1527 (الثالثة وسبعة وعشرين دقيقة بعد الظهر) .

البرنامج التالى يوضح منطق البحث فى جدول المواعيد للعثور على القطار الذى يغادر بعد الساعة 1200 .

البرنامج

قطار فى موعد الغذاء

```
program lunchtrain(input,output);
const startoflunchhour = 1200;
var depart, arrive : integer;
begin
  repeat
     read(depart, arrive)
  until depart >= startoflunchhour;
   write('your train leaves at ', depart);
   writeln(' and arrives at ', arrive)
end.
```

مدخلات هذا البرنامج هى :

موعد القيام	depart
موعد الوصول	arrive

كما يتضمن البرنامج الثابت المسمى startoflunchhour بمعنى « موعد بدء تقديم وجبة الغذاء » ! .

وقد تم التعبير عن الشرط المطلوب بالتعبير المنطقى :

depart >= startoflunchhour

أى أن موعد المغادرة يجب أن يكون بعد (أو يساوى) موعد بدء تقديم وجبة الغذاء .

أما منطق البحث فهو يبدأ بالكلمة **repeat** ، ويليها العمليات المطلوب تكرارها وهى :

● **قراءة موعد القيام والوصول .**

● تطبيق الشروط المذكورة على موعد القيام .

وتنتهى العمليات المتكررة بموجب تحقق الشرط المطلوب وعندئذ تتولى الكلمة until والشرط المصاحب لها بإخراج البرنامج من حلقة البحث إلى المتسلسل الطبيعى له حيث يقوم بطبع المطلوب .

وكما نرى أن العبارة repeat بأجزائها تمثل وحدة مستقلة بداخل البرنامج تبدأ بكلمة repeat وتنتهى بالشرط المقترن بكلمة until .

ولو افترضنا أن العبارات قد تعددت بداخل هذه الوحدة المستقلة فهى لا تحتاج لعزلها عن بقية البرنامج باستخدام begin ، end حيث أنها وحدة مستقلة بذاتها .

ويمكن التعبير عن صيغة العبارة repeat بالآتى :

repeat

statements

(العبارة المعبرة عن العملية
المطلوب تكرارها مفصولة
عن بعضها البعض بفاصلة منقوطة)

until **condition**

(تعبير شرطى)

ملاحظة : فى هذا البرنامج تم إدخال بيانات القيام والوصول يدوياً (من لوحة الأزرار) أى إدخال ثمانية أعداد كل منهم يتكون من أربعة أرقام . وبالطبع فإن الأمور لا تجرى بهذه الطريقة وإلاّ كان النظر فى جدول المواعيد المعلق فى محطة القطارات أسهل بكثير .

إن البيانات يجب أن تكون مخزنة على القرص المغنطيسى (disk) فى ملف كما

ذكرنا من قبل ويتولى الكومبيوتر قراءة المواعيد آلياً بمجرد الضغط على زر تشغيل البرنامج . ولكن فى المرحلة الحالية فإننا نكتفى بالبيانات المدخلة من لوحة الأزرار حتى نتقدم أبدع من ذلك فى قواعد اللغة .

مثال (٤ ــ ١٦) تجميع بيانات تجربة معملية :

فى أحد التجارب يتم تجميع القراءات فى أيام متعاقبة بحيث تُجمع القراءات ، والأيام يوماً بعد يوم حتى يزيد مجموع القراءات عن حد معيّن . عند هذا الحد يتوقف البرنامج ليعطى النتيجة التالية :

« تعدّى مجموع القراءات الحد « كذا »

وقد تم ذلك خلال مدة « كذا » يوماً » .

وهذا هو البرنامج :

البرنامج

بيانات تجربة معملية

```
program countdays(input,output);

const threshold=100;

var nextreading, total :real;
    days :integer;

begin
   days := 0;  total := 0;

   repeat
      days := days + 1;
      read(nextreading);
      total := total + nextreading
   until total > threshold;

   writeln('threshold total of ', threshold,
         ' has been exceeded.');
   writeln('this occurred after ', days, ' days.')
end.
```

شكل (٤ ــ ٢٦)

● **فى هذا البرنامج تم تعريف الحد الذى تتوقف عنده عملية التجميع بالثابت المسمى** threshold وقد تم منحه القيمة العددية **100** .

● **أما متغيرات البرنامج فهى :**

القراءات التالية	nextreading
مجموع القراءات	total
عدد الأيام	days

● **وقد بدأ البرنامج بمنح المتغيرات القيمة صفراً لتفريغ أوعية الجمع من أى محتويات سابقة وذلك بالعبارتين :**

days: =0;total: = 0;

● **بعد ذلك تم تكرار العملية الآتية بالعبارة** repeat :

١ ــ زيادة عدد الأيام بمقدار واحد :

days: = days + 1;

فلو بدأنا بالقيمة « صفراً » فإن ناتج هذه العملية يؤدى إلى منح المتغير days القيمة 1 .

(فالعبارة السابقة يتم قراءتها كالآتى : « أضف واحداً إلى قيمة المتغيّر days وخصص الناتج للمتغير days كقيمة جديدة !) .

٢ ــ أخذ « القراءة التالية » بموجب العبارة read كالآتى :

read(nextreading);

٣ ــ جمع « القراءة التالية » (بعد قراءتها بالطبع) إلى مجموع القراءات السابقة total بالعبارة الآتية :

total: = total + nextreading;

٤ ــ اختبار شرط التوقف كالآتى :

until total<threshold;

وبالخروج من حلقة التكرار والاختبار يتم طبع النتائج المطلوبة بعبارة الطباعة .

(٤ ــ ٩ ــ ٢) عبارة التكرار **while-do** :

تمدنا لغة باسكال بعبارة أخرى للتكرار الذى يتضمن الشرط وهى العبارة while . والفارق الأساسى بين هذه العبارة والعبارة repeat التى عرضناها فى الفقرة السابقة أنه مع العبارة repeat يتم إجراء الاختبار على الشرط بعد تنفيذ العملية المطلوبة ، أما مع العبارة while فيتم اختبار الشرط أولاً .

مثال (٤ ــ ١٧) :

إذا أردنا بناء برنامج تجميع قراءات التجربة الذى عرضناه فى الفقرة السابقة باستخدام هذه العبارة الجديدة فإنه يصبح كالآتى :

```
while total <= threshold do
begin
   days := days + 1;
   read(nextreading);  total := total + nextreading
end
```

Between between

شكل (٤ ــ ٢٧)

ويتضح لنا من المثال أن العبارة تبدأ بكلمة while يعقبها الشرط المطلوب يعقبها كلمة do ثم تبدأ بعد ذلك عملية المعالجة وهى قراءة البيانات وتجميعها . وتنتهى الحلقة بعدم تحقق الشرط فينتقل البرنامج إلى العبارات التالية . ويمكن

التعبير عن الحلقة while بالألفاظ كالآتى : « طالما يسرى هذا الشرط التالى أجر العملية التالية وإلاّ فانتقل إلى العبارات التالية » .

فما الفارق بين العبارتين repeat ، while ؟ .

إن العبارة while تؤدى نفس العمل الذى تؤديه العبارة repeat ، ولكن يتضح لنا من المثال السابق فرق جديد بينهما وهو ضرورة استخدام begin ، end لحصر العبارات **المتعددة** بداخل حلقة التكرار .

ولكن السؤال الذى ربما يخطر لنا الآن هو : ما هى الخاصية التى تتميز بها هذه العبارة على العبارة repeat ؟ وهذا سؤال فى محله .

فبرغم أن المبرمج حر فى اختياره للأداة التى يفضلها لكنه أمام تطبيقات معينة يجب عليه اختيار الأداة المناسبة ولنضرب . هذا المثال البسيط . لنفترض أنك اقترضت مبلغاً من المال من أحد أصدقائك على أن تقوم بتسديده . فى صورة أقساط شهرية . فإذا أردت أن تجعل الكومبيوتر يذكرك كل شهر بما تبقى عليك من الدين فإنك تبرمج هذه العملية الحسابية البسيطة بلغة باسكال كالآتى :

dept := debt – payment

حيث يدل المتغير debt على الدين المتبقى .

ويدل المتغير payment على القسط الشهرى المدفوع أما العملية الحسابية فهى عبارة عن طرح القسط من الدين وتخصيص الناتج للمتغير debt فى الطرف الأيسر والذى يمثل القيمة الجديدة للدين .

فإذا كان الدين يتم تسديده على ١٢ شهراً مثلاً فيمكنك أن تحدد عدد مرات تكرار العملية بحلقة تكرارية ذات عدّاد فتشاهد على الشاشة قيمة الأقساط الشهرية والديون المتبقية خلال العام . أما إذا لم تكن مدة الوفاء بالدين معروفة فإنك تُدخل إلى البرنامج قيمة القسط الشهرى كل مرة فيوافيك بالمبلغ المتبقى .

وسوف تستخدم البرنامج مرة بعد مرة ، ويتناقص الدين عند الدفع كل قسط حتى تصل إلى مرحلة تجد فيها الدين عدداً سالباً ومعناها أن الدين قد انتهى تسديده من الشهر السابق !!

ورغم أن هذا لا يتفق مع واقع الأمور تماماً ، لكن البرنامج الجيد البناء لا يجب أن يستمر فى إعطاء النتائج التى بلا معنى . إذن فهذا البرنامج يحتاج لإضافة شرط ما .

ولكن أين نضع هذا الشرط ؟

لو أننا سوف نستخدم العبارة الشرطية IF فمن الطبيعى أن يتم اختبار المبلغ المتبقى من الدين قبل إجراء أية عملية حسابية والشرط المطلوب هو أن يكون الدين أكبر من القسط !

فإذا أردت طباعة بيانات سداد الدين على مدار عام أو عامين فبصرف النظر عن عدد مرات تكرار الحلقة فإن الشرط الذى يسبق العملية سوف يتولى إيقاف البرنامج فى الوقت المناسب حتى لا يطبع أعداداً سالبة .

والذى يناظر هذا المنطق تماماً هو استخدام الحلقة while فهى تحتوى على الشرط سابقاً للعملية . أما إذا استخدمنا الحلقة repeat فإنها يمكن أن تؤدى إلى وجود أعداد سالبة فى النتائج لأن الاختبار لا يتم إلاّ بعد إجراء العملية .

مثال (٤ ــ ١٨) :

ويمكننا الخروج من هذا المثال البسيط إلى حالة أكثر تعقيداً عندما تصبح المسألة مشاركة فى تجارة ففى هذه الحالة يتم دفع نسبة من الربح علاوة على القسط الشهرى فإذا مثلنا نسبة الربح بالمتغير monthlyrate فإن البرنامج المطلوب يصبح كالآتى :

```
program completestatement(input,output);

var debt, monthlyrate, payment :real;

begin
   read(debt, monthlyrate, payment);
   while debt + debt*monthlyrate/100 >= payment do
   begin
      debt := debt + debt*monthlyrate/100 - payment;
      writeln('debt after next payment is  ', debt)
   end;
   writeln;
   writeln('final payment required will be  ',
           debt + debt*monthlyrate/100)
end.
```

شكل (٤ ــ ٢٨)

وهذا البرنامج يطبع الكشف المطلوب للسداد باستخدام حلقة تكرارية طالما أن الدين **أكبر من أو يساوى** فإذا قلّ عن ذلك خرج البرنامج من الحقة التكرارية ليطبع القيمة المتبقية من الدين والتى تمثل آخر قسط . وهذه العبارة الأخيرة تصبح ذات أهمية عندما لا يكون مبلغ السداد ثابتاً .

مثال (٤ ــ ١٩) اختبار فى الحساب :

من البرامج الهامة التى نحرص على أن تتضمنها كتب اللغات عموماً هى البرامج التعليمية أو التى يطلق عليها الآن التعليم بمعاونة الكومبيوتر Computer Asisted Learning (CAL) وكما نعلم أن هذه النوعية من البرامج بخلاف فائدتها فى التعليم فهى يمكن أن تتطور لتستوعب أرقى فنون البرمجة لتصبح لعبة من ألعاب الكومبيوتر .

والمثال الذى نعرضه الآن لا يُعدّ سوى نواة صغيرة لبرنامج يختبر قدرة

التلميذ على أداء عمليات الجمع كلنه يمكن تطويره تدريجياً ليصل إلى أى درجة من التعقيد .

```
program arithmetictest(input,output);

const a = 16;  b = 25;

var nooftries, answer :integer;

begin
   writeln('lets test your arithmetic.');
   writeln('type the answer to the following sum.');
   write(a, ' + ', b, ' = ');
   read(answer);  nooftries :=1;

   while answer <> a+b do
   begin
      writeln;  writeln('wrong - try again.');
      write(a, ' + ', b, ' = ');
      read(answer);  nooftries := nooftries + 1
   end;

   writeln;
   if nooftries = 1 then
      writeln('very good, got it in one!')
   else writeln('got it at last!');
   writeln('bye for now!')
end.
```

البرنامج
اختبار في الحساب

شكل (٤ ــ ٢٩)

عند تشغيل البرنامج سوف يعرض عليك مسألة جمع هى :

16 + 25

وعليك بكتابة الإجابة باستخدام لوحة الأزرار فإذا كانت الإجابة صحيحة جاءتك رسالة تهنئة حارة :

very good, got it in one !

وإذا كانت الإجابة خاطئة جاءتك الرسالة الآتية تمنحك فرصة محاولة أخرى :

wrong – try again

فإذا كانت المحاولة الثانية ناجحة جاءت التهنئة أقل حرارة من ذى قبل :

got it at last !
bye for now !

والغرض من هذا البرنامج هو عرض كيفية استخدام حلقة التكرار المشروطة فى تكرار المحاولات عند استقبال إجابة خاطئة ، واستخدام العبارة الشرطية لطبع الرسائل المناسبة فى كل حالة . وأولى المطالب اللازمة لتطوير البرنامج هى تغيير الأعداد التى يتم جمعها كل مرة ومن الأفضل أن تكون أعداداً عشوائية تتغير مع كل تنفيذ للبرنامج .

(٤ ـ ٩ ـ ٣) مقارنة بين الحلقتين while ، repeat :

الحلقة while	الحلقة repeat
من الجائز ألاّ تنفذ العبارات بداخل الحلقة على الإطلاق .	تنفذ العبارات بداخل الحلقة مرة واحدة على الأقل .
طالما كان شرط الحلقة نافذاً (true) يستمر الكومبيوتر فى التكرار .	عندما يتحقق شرط الحلقة (true) يتوقف الكومبيوتر عن التكرار .
إذا احتوت حلقة التكرار على أكثر من عبارة لزم استخدام begin فى البداية و end فى النهاية .	ما بين الكلمتين repeat ، until يمكن استخدام أى عدد من العبارات .

(٤ — ١٠) طرق مختلفة لمعالجة البيانات بالحلقات التكرارية :

عندما يكون عدد البيانات المطلوب إدخالها ومعالجتها معروفاً مسبقاً فإن الحلقة for ذات العدّاد تفى بالغرض . حيث يُضبط العدّاد على القيمة المطلوبة ويتم إدخال البيانات ومعالجتها بالتتابع حتى يصل العدّاد إلى قيمته النهائية فيتوقف البرنامج .

لكن هذا ليس هو الحال دائماً . فمع الكثير من أنواع البيانات لا يكون العدد معروفاً مبدئياً . فى مثل هذه الأحوال نستخدم الحلقة repeat أو الحلقة while ونصطنع شرطاً معيناً لإنهاء الحلقة . فإذا كان البرنامج يستقبل أعداداً موجبة فلا بأس من استخدام عدد سالب عندما نريد إنهاء الحلقة التكرارية على أن يتضمن البرنامج الشرط الخاص بإنهاء الحلقة عند وجود عدد سالب .

مثال (٤ — ٢٠) أخطاء شهيرة :

يقوم هذا البرنامج بقراءة وجمع مجموعة من الأرقام الحقيقية الموجبة التى ندخلها إليها واحداً تلو الآخر ، فإذا أدخلنا إليه عدداً سالباً توقف البرنامج عن القراءة والجمع وطبع النتيجة .

```
program add(input,output);

var sum, next :real;

begin
   sum := 0;

   read(next);
   repeat
      sum := sum + next;
      read(next)
   until next < 0;

   writeln('sum is ', sum)
end.
```

شكل (٤ — ٣٠)

ونلاحظ فى هذا المثال أن **القيمة التالية** للعدد المُدخل يتم اختبارها دائماً قبل معالجتها (جمعها على ما سبق من قيم) ، ولذلك لزم قراءة القيمة الأولى قبل الدخول فى الحلقة التكرارية مما استلزم استخدام العبارة read مرتّين ، مرة قبل دخول الحلقة ومرة بداخل الحلقة .

ولعله من الأفضل لاستيعاب هذه الملاحظة أن نعرض بعض الأخطاء الشهيرة التى يمكن أن يقع فيها المبتدئون عند إنشاء مثل هذا البرنامج :

الخطأ الأول :

```
sum := 0;
repeat
   read(next);
   sum := sum + next
until next < 0
```

شكل (٤ ــ ٣١)

يختلف هذا البرنامج عن السابق فى أن عملية المعالجة (الجمع) تسبق عملية اختبار الشرط مما ينتج عنه إضافة القيمة السالبة إلى المجموع الكلى قبل توقف البرنامج .

الخطأ الثانى :

```
sum := 0;
read(next);
repeat
   read(next);
   sum := sum + next
until next < 0
```

شكل (٤ ــ ٣٢)

فى هذا البرنامج لن يتم معالجة أول عدد مُدخل حيث أن العبارة read المخصصة لقراءة العدد الأول تلتها عبارة read التى تقع بداخل الحلقة . كما أن القيمة السالبة سوف تضاف إلى المجموع أيضاً .

الخطأ الثالث :

```
sum := 0;
repeat
   read(next);
   sum := sum + next;
   read(next)
until next > 0
```

شكل (٤ ــ ٣٣)

فى هذا البرنامج سوف يتم جمع القيم الأولى والثالثة والخامسة .. إلى آخره ، وسوف يتم اختبار القيم الثانية والرابعة والسادسة ... إلى آخره ! .

مثال (٤ ــ ٢١) معالجة الانتخابات بالكومبيوتر :

ربما لن تخرج عملية جمع أصوات الناخبين عن العملية السابقة لإضافة مجموعة من الأعداد المدخلة إلى الكومبيوتر بالتتابع ، فيما عدا أن الأعداد التى نقوم بجمعها هذه المرة تنتمى عادة إلى أكثر من فئة حيث تمثل كل فئة حزباً معنياً من الأحزاب المرشحة .

ولنبدأ بفرز نتائج انتخابات حزبين فقط ، ولنفترض أن مهمة البرنامج هى تجميع الأصوات التى حصل عليها كل حزب فى اللجان الانتخابية المختلفة . فى هذه الحالة سوف يكون لدينا فئتان منفصلتان من الأعداد ، حيث يتم جمع عناصر كل فئة على حدة ، كما يجب أيضاً فى هذه الحالة استخدام نهاية مستقلة لكل فئة من البيانات ، بمعنى إدخال عددين سالبين عند انتهاء البيانات .

وهذا هو البرنامج :

البرنامج

معالجة الانتخابات

```
program election(input,output);

var party1next, party2next,
    party1overall, party2overall :integer;

begin
   party1overall := 0;  party2overall := 0;

   read(party1next, party2next);
   repeat
      party1overall := party1overall + party1next;
      party2overall := party2overall + party2next;
      read(party1next, party2next)
   until party1next < 0;

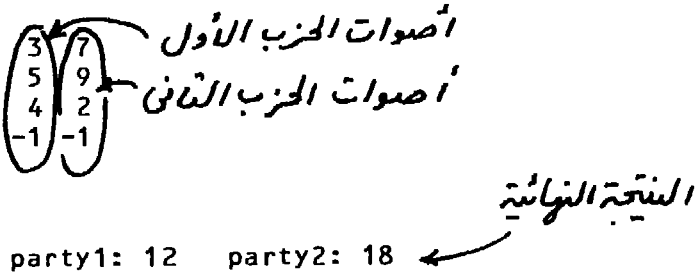
   writeln('party1: ', party1overall, '
          'party2: ', party2ovarall)
end.
```

شكل (٤ ــ ٣٤)

فى هذا البرنامج تم استخدام المتغيرات الآتية :

- party1next مجموع أصوات الحزب الأول فى أى لجنة .
- party2next مجموع أصوات الحزب الثانى فى أى لجنة .
- party1overall مجموع أصوات الحزب الأول الكلية .
- party2overall مجموع أصوات الحزب الثانى الكلية .

وعند تشغيل البرنامج فإنه ينتظر إدخال قيمتين لكل من أصوات الحزبين فى لجنة ما ، وتفصل القيمتان بمسافة خالية ، وعند الضغط على الزر ENTER (أو RETURN) ينتظر إدخال قيمتين جديدتين وهكذا حتى ندخل له عددين سالبين فيتوقف عن الجمع ويطبع النتيجة الممثلة لمجموع أصوات كل حزب على حدة . وهذا مثال لتنفيذ البرنامج :



شكل (٤ ــ ٣٥)

ولعلنا نلاحظ فى هذا البرنامج أنه بالرغم من إدخال عددين سالبين لإنهاء العملية التكرارية لكن الاختبار يتم على عدد واحد منهما فقط ذلك هو العدد الأول المناظر للمتغير party1next . لكنه من الضرورى أن ندخل عددين سالبين لأن تصميم البرنامج يستدعى ذلك بموجب العبارة :

untill party1next<0;

فإذا رغبنا فى أن ندخل قيمة سالبة واحدة كنهاية للبيانات لزم ذلك إجراء تغيير طفيف فى تصميم البرنامج حيث يتم قراءة أصوات الحزب الثانى فى بداية الحلقة وقراءة أصوات الحزب الأول قبل الشرط مباشرة وفى هذه الحالة يمكن إجراء الاختبار على متغير أصوات الحزب الأول فقط كما فى الشكل (٤ ــ ٣٦) .

```
read(party1next);
repeat
   read(party2next);
   party1overall := party1overall + party1next;
   party2overall := party2overall + party2next;
   read(party1next)
until party1next < 0
```

شكل (٤ ـــ ٣٦)

وهناك طريقة أخرى لتصميم البرنامج بحيث يتم إدخال بيانات كل حزب على حدة يعقبها العدد السالب الذى ينهى إدخال البيانات .

وهذه الطريقة أكثر مرونة لا سيما إذا كان أحد الحزبين غير ممثل فى أحد الدوائر وسنترك هذه المعالجة كتمرين على أن نقدم الحل فى نهاية الكتاب .

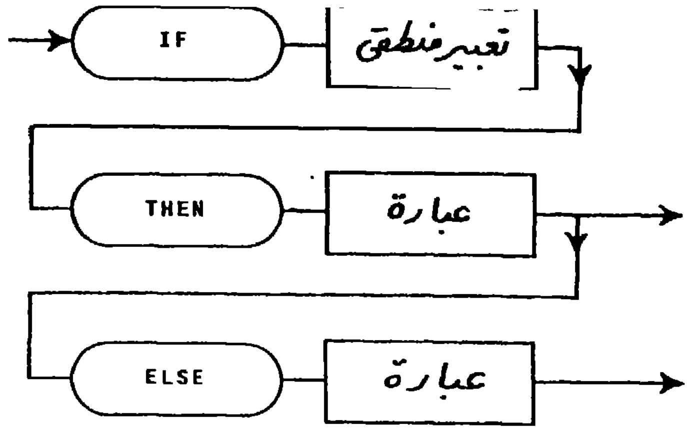


ملخص :

نلخص فيما يلي القواعد التي وردت في هذا الباب بالرسم ولا يفوتنا تذكر المصلحات الآتية التي عرضناها في الأبواب السابقة :

عبارة	statement
تعبير	expression
تعبير منطقي (أو بولياني)	boolean expression
اسم بيان	identifier
متغيّر	variable identifier
ثابت	constant
صحيح	integer
حقيقي	real
عدد صحيح بدون إشارة	unsigned integer
عنوان	label

ونلاحظ في الرسومات التالية أن الكلمات المكتوبة بالحروف الكبيرة هي كلمات من لغة باسكال مثل FOR , DO ,..... أما الكلمات المكتوبة باللغة العربية فهي كلمات ذات معنى (كالمصطلحات السابق ذكرها) وهي تستبدل بما يناسبها من بيانات أو تعبيرات أو متغيرات ... إلخ .



شكل (٤ ــ ٣٧)
العبارة الشرطية (IF)

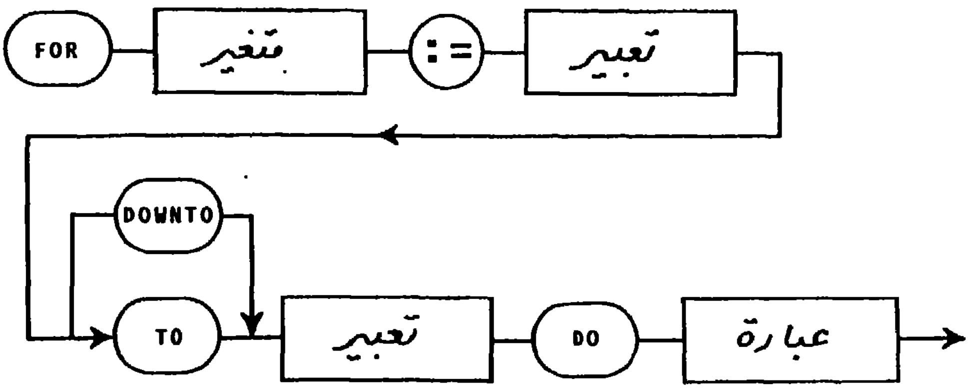
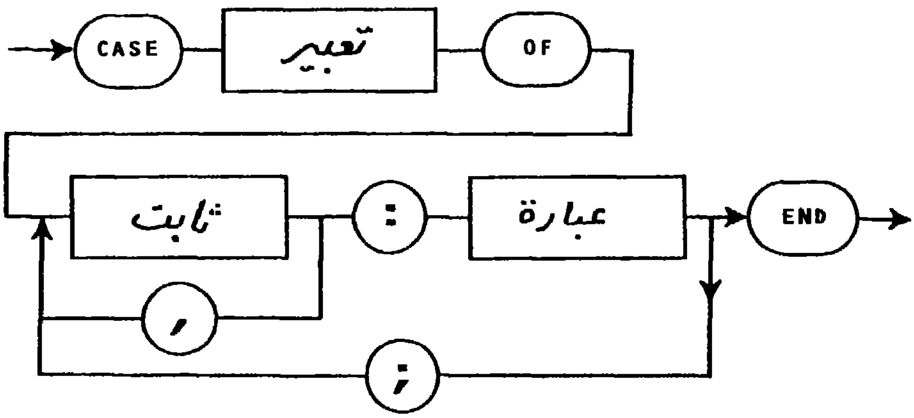
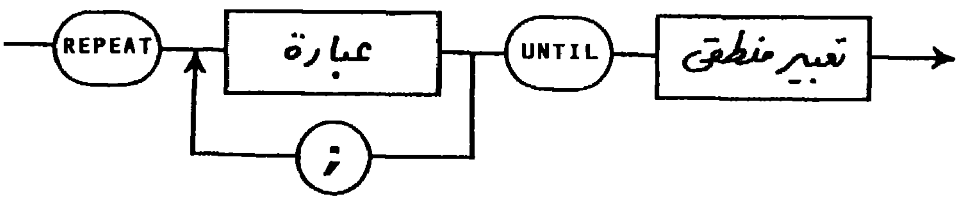


Fig. 6.2

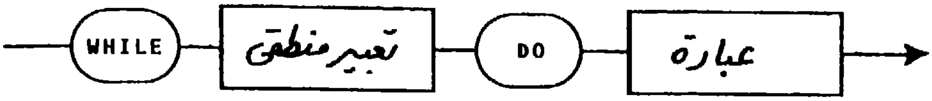
شكل (٤ ــ ٣٨)
عبارة التكرار (for)



شكل (٤ ـــ ٣٩)
عبارة الحالات المتعددة (case)



شكل (٤ ـــ ٤٠)
عبارة التكرار المشروط (repeat)



شكل (٤ ــ ٤١)
عبارة التكرار المشروط (while)



■ تمرينات الباب الرابع :

س (٤ ــ ١) اكتب برنامجاً يطبع جدول الضرب للأعداد المختلفة كالمثال الآتى :

```
5
1x5=5
2x5=10
3x5=15
4x5=20
5x5=25
6x5=30
7x5=35
8x5=40
9x5=45
10x5=50
```

شكل (٤ ــ ٤٢)

س (٤ ــ ٢) ينتج عن استخدام العبارة :

writeln('*:i)

طبع نجمة واحدة متبوعة بعدد **(i − 1)** مسافة خالية . والواقع أنه يجوز أن يستبدل المتغير i بأى تعبير صحيح (integer) إذا دعت الحاجة .

اكتب برنامجاً يقرأ قيمة متغير صحيح n ويطبع الرقم 7 باستخدام النجوم فى عدد n من السطور . وعلى سبيل المثال فلو كان n=6 فإن الخرج يصبح كالتالى :

```
******
    *
   *
  *
 *
*
```

شكل (٤ ــ ٤٣)

س (٤ ــ ٣) اكتب برنامجاً بلغة باسكال لرسم مثلث يحتل عدد n من السطور حيث n هو متغير صحيح يقوم البرنامج بقراءته عند التنفيذ .

وعلى سبيل المثال إذا كانت n=7 فإن الخرج يكون فى شكل (٤ ــ ٤٤) .

```
      *
     * *
    *   *
   *     *
  *       *
 *         *
*************
```

شكل (٤ ــ ٤٤)

س (٤ ــ ٤) المطلوب إعداد برنامج لمحل بيع للأحذية يستخدم المقاييس البريطانية ، لكى يرشد البائع إلى مقاس الحذاء بالنظام الأمريكى إذا اقتضى الأمر .

والعلاقة بين النظامين ليست علاقة خطية كما أنها تختلف بحسب ما إذا كان الحذاء « حريمى » أو « رجالى » .

وهذا نموذج للخرج المطبوع من البرنامج المطلوب :

<u>Men's</u> <u>shoes</u> الأحذية الرجالي

British	7	8	9	10	11
American	7.5	8.5	9.5	10.5	11.5

<u>Women's</u> <u>shoes</u> الأحذية الحريمي

British	3	4	5	6	7
American	4.5	5.5	6.5	7.5	8.5

شكل (٤ ــ ٤٥)

أما المدخلات للبرنامج فهى مقاس الحذاء بالنظام البريطانى وأحد الرقمين 1 أو 0 لتمثيل الحذاء الرجالى أو الحريمى .

يمكنك إضافة المقاييس الفرنسية أيضاً لهذا البرنامج .

س (٤ ــ ٥) اكتب برنامجاً يقرأ العدد الممثل للشهر مثل ...,jan, feb, mar ويطبع عدد أيام هذا الشهر ، وذلك بفرض أن السنة غير كبيسة .

س (٤ ــ ٦) بتطوير البرنامج السابق يمكن أن يقوم البرنامج أيضاً بكتابة اسم الشهر مستدلاً عليه برقمه كالمثال الآتى :

```
please enter month number
12
 december.. this month contains: 31 days

>
```

شكل (٤ ــ ٤٦)

س (٤ ــ ٧) بتطوير البرنامج السابق اجعل البرنامج يقرأ السنة علاوة على شهر ويقوم باختبار السنة إذا ما كانت كبيسة أم لا ثم يطبع عدد أيام الشهر .

(ملاحظة : فبراير فى السنة الكبيسة ٢٩ يوماً

فبراير فى السنة غير الكبيسة ٢٨) .

س (٤ ــ ٨) فى أحد معاهد الكومبيوتر يتم ترقيم المناهج التى تدرس بالمعهد بالأرقام من 1 إلى 10 ، ويتم أداء المحاضرات فى فترات قدرها ساعتين لكل محاضرة .

والجدول الآتى بعد يمثل المواعيد المختلفة لهذه المناهج والمحاضرات حيث يوضح الجدول رقم المنهج (course 1,2,...) واليوم (thu, mon,. .) والساعة : (9am, 10am...)

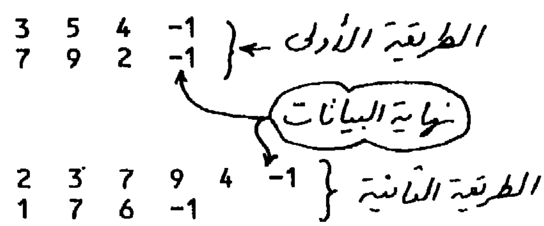
Course	1,2	thu	9am	fri	10am
Course	3	mon	10am	thu	10am
Course	4,5	mon	11am	tue	11am
Course	6,7	tue	9am	wed	2pm
Course	8	mon	12am	thu	9am
Course	9	tue	10am	wed	11am
Course	10	fri	9am	fri	11am

شكل (٤ ــ ٤٧)

اكتب برنامجاً يمكن استخدامه للاستعلام عن مواعيد المحاضرات حيث يقرأ البرنامج رقم المنهج ويطبع رسالة بالمواعيد المحددة للمحاضرة (اليوم والساعة) خلال الأسبوع .

س (٤ ــ ٩) فى المثال السابق إذا عبرنا عن أيام الأسبوع بالأرقام من 1 إلى 6 ، وساعات اليوم بالأرقام من 1 إلى 6 (بدءاً من الساعة ٩ صباحاً (9am) وحتى الثانية بعد الظهر (2pm)) . فإنه يمكن تمثيل الفترة الزمنية للمحاضرة بعدد واحد ذى رقمين الرقم الأول يعبر عن اليوم والرقم الثانى يعبر عن الساعة . معنى ذلك أن العدد 12 يعنى يوم السبت (1) الساعة العاشرة صباحاً (2) . اكتب برنامجاً يستقبل هذا الكود الممثل لليوم والساعة ويطبع على الشاشة أية محاضرات تعقد فى هذا الموعد .

س (٤ ــ ١٠) اكتب برنامجاً لجمع الأصوات التى يحصل عليها حزبان فى الدوائر المختلفة على أن يتم تشغيل البرنامج بأحد الطرق الآتية :



شكل (٤ ــ ٤٨)

وكما نرى فإنه مع الطريقة الثانية فإن الحزب الثانى غير ممثل فى بعض الدوائر الانتخابية لذلك فقد انتهت بياناته قبل الآخر .

الباب الخامس

منشآت التحكم
(CONTROL STRUCTURES)

مفتتح

إذا شرعت فى برمجة مشكلة ذات أبعاد حقيقية فإنك سوف تحتاج لبعض المهارات فى إنشاء البرنامج خاصة عندما تتشابك العلاقات التى تربط بين المتغيرات فى الأجزاء المختلفة للبرنامج .

فقد يتطلب الأمر فى بعض الأحيان أن تحتوى حلقة تكرارية على عبارة شرطية بداخلها أو تحتوى العبارة الشرطية على حلقة تكرارية أو أن تتداخل الحلقات التكرارية بداخل بعضها البعض .

هذه التراكيب المختلفة تسمى منشآت التحكم وهى وإن كانت تتطلب مهارات معينة فى المعالجة لكنها مع ذلك خبرات معروفة يمكن اكتسابها من خلال الأمثلة التى نعرضها فى هذا الباب والتى تعالج مشكلات حقيقية .

(٥ ــ ١) الحلقات التكرارية المحتوية على عبارات شرطية :

علمنا من قبل أن الحلقة التكرارية for تتبع الصيغة الآتية :

for variable: = expression to expression do statement

والعبارة المطلوب تكرارها (statement) قد تكون على أية صورة من صور عبارات باسكال التى درسناها . وإحدى الصور الممكنة هى العبارة الشرطية . ولنر مثالاً لهذه الحالة :

مثال (٥ ــ ١) معالجة رياضية :

إذا أعطيت عدداً صحيحاً مثل 18 فإنه يمكن إيجاد الآعداد التى يقبل القسمة عليها بدون باق والتى تنحصر بين 2 ، 9 بطريقة يدوية ، وهى الأعداد :

2 , 3 , 6 , 9

فإذا أردنا أن نوكل هذا العمل إلى الكومبيوتر فإن دالة الباق MOD التى تعرفنا بها من قبل يمكن أن تساعدنا فى برمجة الحل ، حتى يكون أكثر عمومية .

فلو عبرنا عن العدد المقسوم (18) بالمتغير giveninteger ولو عبرنا عن الأعداد المقسوم عليها بالمتغير i فإنه لجميع القيم المطلوبة (مثل 2 ، 3 ، 6 ، 9) يكون :

giveninteger MOD i = 0

هذا هو شرط القسمة بدون باق .

والبرنامج التالى يقرأ عدداً صحيحاً ويستقبله فى المتغير giveninteger ثم يستخدم الحلقة التكرارية (من 2 إلى 9) لإيجاد الأعداد الصحيحة الموجبة التى تقع فى هذا النطاق والتى يقبل القسمة عليها المتغير giveninteger بدون باق .

والعبارة الشرطية هنا تقع بداخل حلقة التكرار حيث أن الشرط يتم اختباره لكل عدد من الأعداد بين 2 ، 9 .

البرنامج

```
program factors(input, output);

var giveninteger, i : integer;

begin
   read(giveninteger);
   for i := 2 to 9 do
      if giveninteger mod i = 0 then
         writeln(i, ' divides into the given integer.')
end.
```

شكل (٥ ــ ١) ا

والآتى بعد مثال لتنفيذ البرنامج مع **العدد 18** :

التنفيذ

```
2 divides into the given integer.
3 divides into the given integer.
6 divides into the given integer.
9 divides into the given integer.
```

شكل (٥ ــ ١) ب

مثال (٥ ــ ٢) المراقبة الرقمية للإنتاج :

عالجنا فى الباب الرابع فى البرنامج (tolerence2) عملية مضاهاة مجموعة من

الأعداد برقم قياسى بهدف حصر الأعداد التى تقترب قيمتها من قيمة الرقم القياسى .

وقد استخدمنا فى هذا البرنامج العبارة الشرطية بداخل حلقة التكرار .

ولعلنا فى هذا المثال نضيف لمسة واقعية إلى المعالجة المذكورة لتوضيح فائدة مثل هذا البرنامج . ففى مصانع الإنتاج بالجملة عادة ما يستخدم الكومبيوتر فى مراقبة الإنتاج . فإذا كان المصنع يقوم بإنتاج قضبان معدنية ذات أطوال معينة . فإن جهاز القياس الذى يقوم باختبار طول القضبان الخارجة من خط الإنتاج عادة ما يكون مبرمجاً (كومبيوتر ذو غرض خاص) .

وفى المثال الذى نحن بصدده سوف نجرى عملية محاكاة simulation للعملية الصناعية المذكورة . فالأعداد التى نستقبلها فى البرنامج تمثل القياسات الناتجة من جهاز قياس الأطوال . ويقوم البرنامج بحصر الأطوال التى لا تختلف عن الطول القياسى إلاّ فى حدود العُشر . ويتوقف عمل البرنامج عندما يستقبل عدداً سالباً .

وفى هذا البرنامج سوف نستخدم المتغيرات التالية :

- standard ثابت مسمى قيمته 6.37 للتعبير عن طول قياسى معيّن .
- length متغير حقيقى يعبر عن الطول .
- nowithin متغير صحيح لحصر عدد الأطوال التى تقع فى النطاق المقبول .
- nowithout متغير صحيح لحصر عدد الأطوال التى تقع خارج النطاق المقبول

وينتهى البرنامج بطبع قيمتى المتغيرين الأخيرين .

البرنامج

المراقبة الرقمية للإنتاج

```
program tolerance3(input,output);

const standard = 6.37;
var length : real;
    nowithin, nowithout : integer;

begin

   nowithin := 0;
   nowithout:= 0;

   read(length);

   repeat

      if abs(length - standard) < 0.1 then
         nowithin := nowithin + 1
      else
         nowithout:= nowithout + 1;

      read(length)

   until length < 0;

   writeln(nowithin,  ' values are within  tolerance.')
   writeln(nowithout, ' values are outside tolerance.')
end.
```

شكل (٥ ــ ٢)

نتائج ستة تلاميذ فى ست مواد مختلفة فى الامتحان الشهرى ممثلة فى صورة جدول (مصفوفة) .

(٥ ــ ٢) الحلقات التكرارية المتداخلة *nested loops* :

تحتاج بعض التطبيقات إلى استخدام أكثر من حلقة تكرارية « تحتضن » بعضها البعض . وأهم هذه التطبيقات هو الجداول ، مثل الجدول الذى يحتوى

على نتيجة الطلبة فى أحد الكليات فى نهاية العام الدراسى .

ولو أننا أردنا تمثيل درجات طالب واحد فى ثمانى مواد دراسية على سبيل المثال ، فإن برنامج القراءة سوف يكرر عملية القراءة ثمانى مرات . فإذا كان البرنامج يقرأ درجات عشرة طلبة فإننا نحتاج لحلقتين تكراريتين الأولى تتغير مع كل طالب (من واحد إلى عشرة) والثانية تتغير مع كل مادة (من واحد إلى ثمانية) لكل طالب من الطلبة العشرة . والجدول الموضح فى الشكل (٥ — ٣) يبين درجات بعض التلاميذ فى صورة جدول . ونرى أن درجات التلميذ الواحد تحتل **صفاً** كاملاً أى أن **الصف** يعبّر عن عدد المواد الدراسية بينما يعبّر **العمود** عن عدد الطلبة .

ومن سبق له دراسة لغة بيسك فإنه لا بد قد صادف مصطلح المصفوفة (matrix) للتعبير عن هذا الجدول .

والمثال الآتى يوضح كيفية قراءة درجات خمس وعشرين تلميذاً فى خمس مواد دراسية وجمع درجات كل تلميذ على حدة لبيان نتيجته ومجموعه . وقد استخدمنا فى هذا البرنامج **حلقة خارجية** ذات عدّاد (من 1 إلى 25) تمثل الطلبة **وحلقة داخلية** ذات عدّاد من (1 إلى 5) تمثل المواد أى أن الحلقة الداخلية تتغير خمس مرات مع كل قراءة من قراءات عدّاد الحلقة الخارجية أى أنها تنفذ 5×25 مرة !

المواد / الأسماء	اللغة العربية	التربية الدينية	الجغرافيا والتاريخ	الحساب والهندسة	العلوم والصحة	الرسم
محمد سمير	17	15	18	20	13	8
مروان فاروق	15	12	17	17	14	12
هشام عاشور	20	18	20	19	20	10
حازم أسامة	19	18	20	20	19	13
كريم رفعت	18	19	20	19	20	12
رامى الحسينى	17	19	15	17	17	15

نتائج ستة تلاميذ فى ست مواد مختلفة فى الامتحان الشهرى ممثلة فى صورة جدول (مصفوفة)

شكل (٥ ــ ٣)

مثال (٥ ــ ٣) نتيجة فصل دراسى :

المتغيّرات المستخدمة فى البرنامج هى :

candidate متغير العدّاد للحلقة الخارجية (حلقة عدد الطلبة) وبالطبع يمكن استخدام أى حرف مثل x أو y للتعبير عن المتغير ولكن كلمة candidate تذكرك دائماً بأن هذه هى

حلقة عدد الطلبة عندما تقرأ البرنامج . وهذه خاصية مفيدة من خصائص لغة باسكال حيث تسمح باستخدام المتغيرات ذات الأسماء الطويلة كمتغيرات للعدّادات .

exam متغيّر العدّاد للحلقة الداخلية (حلقة عدد المواد)

total متغير مجموع الدرجات . ولأنه يستخدم كوعاء لتجميع درجات كل طالب فإنه يتم تفريغه (مساواته بالصفر) عند بداية كل دورة من دورات الحلقة الخارجية .

mark متغير يمثل درجة أى مادة من المواد .

average متغير يمثل متوسط مجموع الدرجات وهو يساوى مجموع الدرجات مقسوماً على عدد المواد .

ويعتبر الطالب ناجحاً إذا حصل على متوسط **قدره 50** أو أكثر .

وبالطبع يمكن تطوير هذا البرنامج باستقبال أسماء التلاميذ أيضاً وطبعها فى الخرج حتى تكون النتيجة مقبولة شكلاً . فالبرنامج فى صورته الحالية يطبع أرقام التلاميذ بدلاً من أسمائهم .

```
program examarks(input,output);

var candidate, exam,  total, mark : integer;
    average : real;
begin
   for candidate := 1 to 25 do
   begin

      total  := 0;

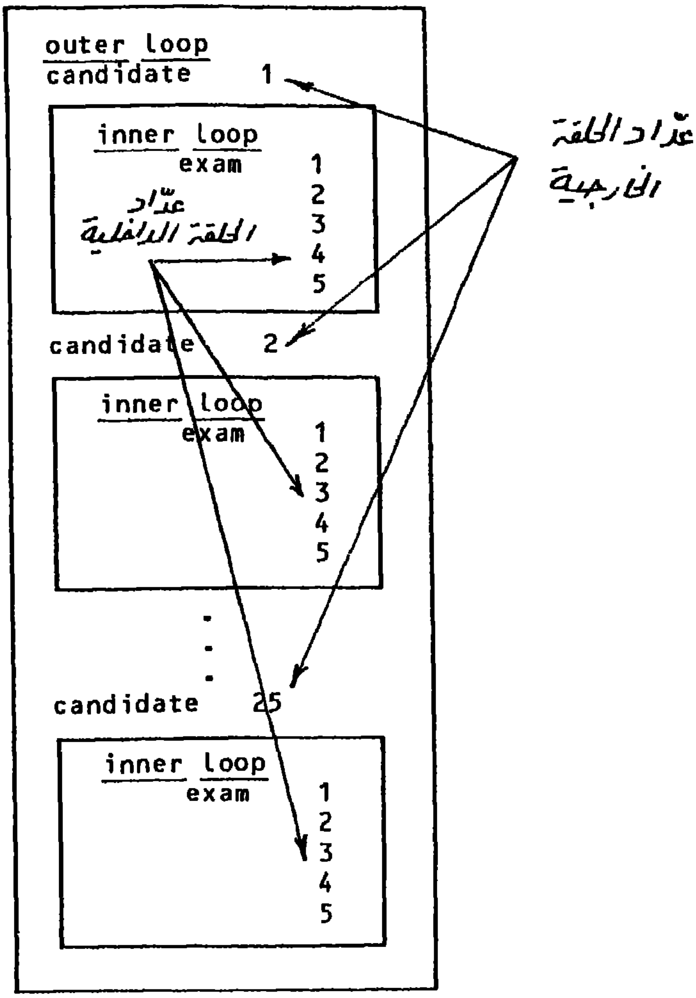
      for exam  := 1 to 5 do
      begin
         read(mark);
         total  := total + mark;
      end;

      average := total/5;
      if average >= 50 then
         writeln('candidate  ', candidate,
                  ' passed, average is ', average)
      else
         writeln('candidate  ', candidate,
                  ' failed, average is ', average)

   end
end.
```

شكل (٥ ــ ٤)

والشكل (٥ ــ ٥) يوضح مجرى الأحداث فى البرنامج . فالحلقة الخارجية تبدأ بالرقم 1 الذى يمثل الطالب الأول (candidate = 1) . ويظل العدّاد محتفظاً بالرقم 1 أثناء تنفيذ الحلقة الداخلية بأكملها والتى تبدأ من (exam = 1) وتتكرر خمس مرات حتى تصل إلى (exam = 5) ممثلة عدد الامتحانات أو عدد المواد الدراسية . وبعد طبع نتيجة الطالب الأول تتغير قيمة العدّاد للحلقة الخارجية لتصبح 2 ممثلة الطالب الثانى وهكذا حتى الطالب الخامس والعشرين .



شكل (٥ ـــ ٥)
كيف تجرى الأحداث فى البرنامج

مثال (٥ ـــ ٤) الرسم بالحلقات التكرارية المتداخلة :

قدمنا من قبل بعض أمثلة وتدريبات على الرسم لكن استخدام الحلقات

التكرارية المتداخلة تمكننا من رسم أشكال أكثر تعقيداً كالرسم الموضح شكل (٥ ـــ ٦) .

```
.........
....*....
...***...
..*****..
.*******.
.........
```

شكل (٥ ـــ ٦)

كيف يتم بناء هذا الرسم كمنطق فى البرنامج ؟

يمكننا تمثيل المنطق الذى نرسم به مثل هذا الشكل بالخطوات المنطقية الآتية التى يطلق عليها الكود الكاذب (pseudo code) والتى تكافىء خريطة تسلسل المنطق (flowchart) :

- **اقرأ عدد الصفوف فى الشكل المراد رسمه .**
- **احسب عرض الشكل :**

١ ـــ كرّر الآتى :

- **ارسم نقطة**

حتى تصل إلى آخر عرض الشكل

- **اترك سطراً خالياً**

٢ ـــ كرر الآتى بدءاً من الصف رقم 1 :

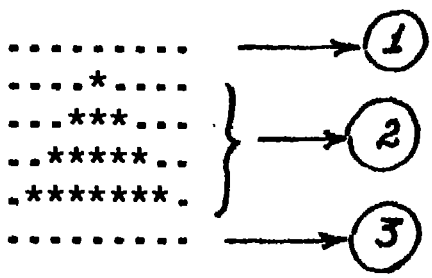
- **ارسم صفاً من صفوف المثلث بالعدد المناسب من النقط على الجانبين حتى تصل إلى نهاية عدد الصفوف .**

٣ ـــ كرر الآتى :

- **ارسم نقطة**

حتى تصل إلى آخر عرض الشكل .

والروتينات ١ ، ٢ ، ٣ المشار إليها تمثل خطوات رسم الأجزاء التى تحمل نفس الأرقام فى الشكل (٥ — ٧) :



شكل (٥ — ٧)

أما عرض الشكل فيمكن حسابه بمعلومية عدد الصفوف على أساس العلاقة التالية :

عرض الشكل = عدد الصفوف * ٢ + ١

أما الجزء الذى يحتاج للتفصيل فهو رسم صفوف المثلث (الجزء رقم (٢) من الشكل) ويمكن وصف ذلك بالمنطق الآتى :

● احسب عدد النجوم فى الصف وليكن (noofstars)

● احسب عدد النقط على جانبى النجوم فى الصف (noofdots)

كرر الآتى :

● ارسم نقطة

حتى تبلغ قراءة العداد القيمة noofdots

كرر الآتى :

● ارسم نجمة

حتى تبلغ قراءة العداد القيمة noofstars

كرر الآتى :

● ارسم نقطة

حتى تبلغ قراءة العدّاد القيمة noofdots

ويمكن التعبير عن عدد النجوم فى الصف المعيّن بالعلاقة :

عدد النجوم = ٢ * رقم الصف ـــ ١

* * (لاحظ أن الصف الأول هنا هو السطر الثانى فى الشكل كله حيث أن الصف الأول هنا معناه الصف الأول من الشكل المثلثى المرسوم بالنجوم) .

.عدد النجوم فى الصف الأول : ٢ * ١ ـــ ١ = ١

النجوم فى الصف الثانى : ٢ * ٢ ـــ ١ = ٣

النجوم فى الصف الثالث : ٢ * ٣ ـــ ١ = ٥ .

وهكذا ..

أما عدد النقط فى كل سطر فيتم التعبير عنها بالعلاقة :

عدد النقط على كَل جانب = عدد الصفوف + ١ ـــ رقم الصنف

ولاختبار هذه العلاقة نضرب الأمثلة الآتية :

عدد نقط الصف الأول = ٤ + ١ ــ ١ = ٤
عدد نقط الصف الثانى = ٤ + ١ ــ ٢ = ٣
عدد نقط الصف الثالث = ٤ + ١ ــ ٣ = ٢

وهكذا ...

وهذا المنطق يجب أن يندمج فى منطق البرنامج الكلى ضمن الروتين الثانى لرسم مثلث النجوم . والشكل رقم (٥ ــ٨) يوضح البرنامج الكامل لرسم الشكل المطلوب وقد استخدمت فيه المتغيرات الآتية :

- noofrows عدد الصفوف (للمثلث فقط)
- widthofpicture عرض الشكل (مقاساً باللبنة)
- ch متغير العدّاد
- rowno رقم الصف
- noofstars عدد النجوم
- noofdots عدد النقط

والبرنامج نفس المنطق الذى سبق عرضه مع ملاحظة إضافة بداية ونهاية للروتين رقم (٢) لاحتوائه على عدد كبير من الخطوات وقد تم وضع الروتين كله داخل برواز للإيضاح .

```
program framedtriangle(input,output);

var noofrows, widthofpicture, ch,
    rowno, noofstars, noofdots : integer;

begin
   read(noofrows);
   widthofpicture := 2*noofrows + 1;
   for ch := 1 to widthofpicture do write('.');
   writeln;

   for rowno := 1 to noofrows do
   begin
      noofstars := 2*rowno - 1;
      noofdots  := noofrows + 1 - rowno;
      for ch := 1 to noofdots  do write('.');
      for ch := 1 to noofstars do write('*');
      for ch := 1 to noofdots  do write('.');
      writeln
   end;

   for ch := 1 to widthofpicture do write('.');
   writeln
end.
```

شكل (٥ ــ ٨)

وعند تنفيذ البرنامج يبدأ باستقبال عدد صفوف المثلث المطلوب رسمه وعلى أساس ذلك يتحدد حجم الشكل كله جرب البرنامج باستخدام أعداداً مختلفة وشاهد النتائج .

(٥ ــ ٣) العبارات الشرطية المتداخلة nested if-statements :

علمنا من قبل أن العبارات الشرطية يجوز أن تقع بداخل حلقة تكرارية منعاً لتكرار كتابة العبارة الشرطية عدة مرات بداخل البرنامج ، ولكن هذه الحالة تنطبق فقط على العبارة الشرطية المتكررة .

كما تعاملنا من قبل مع العبارة الشرطية الكاملة (if-then-else ذات النتيجة

الأصلية والنتيجة البديلة .

ولكن هناك حالات تتعقد فيها الشروط وتتشابك بحيث يضطر المبرمج أن يكتب عدة عبارات شرطية متتابعة لوصف المشكلة كالمثال الآتى :

مثال (٥ ــ ٥) تمثيل البدائل المتعدّدة :

الجدول الآتى يمثل الحالات المختلفة التى يلتقى بها مندوب شركة التأمين وعلى أساسها يحدد نوع التأمين الذى يمكنه منحه للسيارة على أساس سعة الماكينة ، وعمر السائق ، وعدد الحوادث التى وقعت للسيارة من قبل . فالسعة تحدد إذا ما كانت السيارة كبيرة أو صغيرة وبالتالى تدل على ثمنها وتحمّل شركة التأمين مسئولية أكبر من ناحية التعويض . وعمر السائق يصبح عنصراً هاماً للشركة إذا كان يقل عن ٢١ سنة ، فمن المتوقع أن يكون السائق أكثر حباً للمخاطرة وأكثر تعرضاً للحوادث وهذا أيضاً يرفع مبلغ التأمين على السيارة . ومع ذلك فحكم السن ليس نهائياً بل يضاف إليه تاريخ الحوادث التى وقعت للسائق من قبل وثبتت إدانته فيها فإذا كانت أكثر من ثلاثة رفضت الشركة منح بوليصة التأمين نهائياً وهكذا ..

والمطلوب من البرنامج أن يطبع الجدول الموضح على الشاشة :

العمر age	سعة الماكينة engine size	عدد مرات الإدانة convictions	الرسالة على الشاشة message
>=21	>=2000	>=3	policy loaded by 45%
>=21	>=2000	< 3	policy loaded by 15%
>=21	< 2000	>=3	policy loaded by 30%
>=21	< 2000	< 3	no loading
< 21	>=2000	>=3	no policy to be issued
< 21	>=2000	< 3	policy loaded by 60%
< 21	< 2000	>=3	policy loaded by 50%
< 21	< 2000	< 3	policy loaded by 10%

لا تمنح البوليصة — النسبة المئوية للزيادة فى الثمن

شكل (٥ ــ ٩)

أحد الطرق الممكنة لتمثيل هذا الجدول هو البرنامج شكل (٥ ــ ١٠) وهو يستخدم المتغيرات الآتية :

١ ــ المتغيرات المنطقية (boolean) :

over21 : يدل على الشخص الذى يزيد عمره عن أو يساوى ٢١ عاماً
largecar : يدل على السيارة الكبيرة التى تزيد سعتها عن ٢٠٠٠
riskdriver : يدل على السائق المخاطر الذى زادت مرّات إدانته عن ثلاثة .

٢ ــ المتغيرات الصحيحة (integer) :

age : العمر
cc : سعة السيارة
convictions : عدد مرّات الإدانة

والبرنامج عبارة عن ترجمة حرفيّة لسياسة التأمين التى يعرضها الجدول شكل (٥ ــ ٩) لذلك فهو يحتوى على عبارات شرطية بعدد سطور الجدول تماماً .

البرنامج ①

```
program policy(input,output);

var over21 , largecar , riskdriver : boolean;
        age , cc , convictions : integer;

begin
   read(age,cc,convictions);

   over21  := age >= 21;
   largecar:= cc >= 2000;
   riskdriver:= convictions >= 3;

   if over21     and     largecar and     riskdriver
   then writeln('policy loaded by 45 percent');

   if over21     and     largecar and not riskdriver
   then writeln('policy loaded by 15 percent');

   if over21     and not largecar and     riskdriver
   then writeln('policy loaded by 30 percent');

   if over21     and not largecar and not riskdriver
   then writeln('no loading');

   if not over21 and     largecar and     riskdriver
   then writeln('no policy to be issued');

   if not over21 and     largecar and not riskdriver
   then writeln('policy loaded by 60 percent');

   if not over21 and not largecar and     riskdriver
   then writeln('policy loaded by 50 percent');

   if not over21 and not largecar and not riskdriver
   then writeln('policy loaded by 10 percent')

end.
```

شكل (٥ ـــ ١٠)

بالتأكيد هناك طريقة أفضل بل طرق أفضل من هذه الطريقة البدائية التى تستهلك وقت الكومبيوتر .

ومن أفضل الطرق التى يمكن بواسطتها إنشاء برنامج كهذا هى العبارات الشرطية المتداخلة (nested) التى تحتضن بعضها البعض . بمعنى أن نتيجة العبارات الشرطية (سواء الأصلية أو البديلة) يمكن أن تحتوى على عبارة شرطية أخرى . فإذا ألقينا نظرة ثانية على الجدول وعلى خانة « العمر » بالتحديد لوجدناه يعالج حالتين :

(١) إذا كان العمر ٢١ سنة أو أكثر .

(٢) وإذا كان العمر أقل من ٢١ سنة .

يمكن تمثيل هاتين الحالتين بعبارة شرطية واحدة كالآتى :

if over21 then

(الحالات الأربع الأولى ...النتيجة الأصلية)

⋮

else

(الحالات الأربع الثانية .... النتيجة البديلة)

ويلى كلمة then جميع النتائج المترتبة على هذا الشرط (أكبر من ٢١ سنة) ويلى كلمة else جميع النتائج التى تترتب على هذا الشرط (أقل من ٢١ سنة) .

والحالات الأربع الأولى تتضمن شروطاً أخرى خلاف عمر السائق مثل سعة سيارته وعدد مرّات إدانته فى حوادث سيارات . لذلك فالنتيجة الأصلية التى تعقب كلمة then سوف تحتوى على عبارة شرطية أو أكثر ..

وكذلك الحال بالنسبة للنتيجة البديلة التى تعقب العبارة else . أى أن العبارة الشرطية سوف تحتضن بداخلها عبارات شرطية أخرى فتبدو كما فى الشكل (٥ ـــ ١١) ونلاحظ أن الأرقام التى بداخل الدوائر تعبّر عن

الرسالات المختلفة التى تظهر على الشاشة ممثلة الحالات الثمانية بالترتيب الوارد فى الجدول شكل (٥ — ٩) .

```
if over21 then
    if largecar then
        if riskdriver then ....  (1)
        else ...  (2)
    else
        if riskdriver then ...  (3)
        else ...  (4)
else
    if largecar then
        if riskdriver then ...  (5)
        else ...  (6)
    else
        if riskdriver then ...  (7)
        else ...  (8)
```

الحالات الأربع الأولى

الحالات الأربع الثانية

شكل (٥ — ١١)

إن هذه العبارات الشرطية المتداخلة توّفر وقت الكومبيوتر فلو كان طالب البوليصة يخضع للشرط الثامن مثلاً فإنه مع البرنامج شكل (٥ — ١٠) سيقوم الكومبيوتر باختبار صحة الشروط السبعة الأولى حتى يصل إلى الحالة الثامنة التى تنطبق على طالب البوليصة .

أما مع هذا المنطق الجديد شكل (٥ ــ ١١) فإن الكومبيوتر سوف يخرج من الحالات الأربع الأولى بمجرد اختبار واحد فقط وهو اختبار العمر (لأن جميع هذه الحالات تمثل النتيجة الأصلية للعبارة الشرطية) .

فإذا خرجنا إلى النتيجة البديلة التى تعالج الحالات الأربع الثانية لوجدنا أن الحالتين ٥ ، ٦ تختبران بشرط واحد فقط وهو (....if largecar) لذلك يخرج البرنامج من هذا الاختبار إلى الاختبار السابع مباشرة الذى يلى كلمة else .

أى أن عدد الاختبارات التى قام بها البرنامج للوصول للحالة رقم ٨ هو ٣ اختبارات فقط .

بقى أن نكتب مضمون الرسائل الثمانية فى البرنامج حتى تكتمل معالمه . والبرنامج شكل (٥ ــ ٢) يمثل هذه الصور النهائية وقد استخدمنا فيه الثوابت الحرفية للحفاظ على شكل البرنامج « المنطقى » !

البرنامج ٢

```
program policy2(input,output);

const p45 = 'policy loaded by 45 percent';
      p15 = 'policy loaded by 15 percent';
      p30 = 'policy loaded by 30 per cent';
      ok  = 'no loading';
      no  = 'no policy to be issued';
      p60 = 'policy loaded by 60 percent';
      p50 = 'policy loaded by 50 percent';
      p10 = 'policy loaded by 10 percent';
var over21, largecar, riskdriver : boolean;
    age, cc,convictions : integer;

begin
   read(age,cc,convictions);
   over21 := age >= 21;
   largecar := cc >= 2000;
   riskdriver := convictions >= 3;

   if over21 then

      if largecar then

         if riskdriver then writeln(p45)
         else               writeln(p15)

      else

         if riskdriver then writeln(p30)
         else               writeln(ok)

   else

      if largecar then

         if riskdriver then writeln(no)
         else               writeln(p60)

      else

         if riskdriver then writeln(p50)
         else               writeln(p10)

end.
```

وضع الرسائل في ثوابت حرفية

الحالات الأربع الأولى

الحالات الأربع الثانية

وهناك دائماً حل آخر .. فقد استخدمنا من قبل العبارة case لتمثيل البدائل المتعددة وهى بلا شك تصلح لتمثيل هذه الحالة مع استخدام تكنيك جديد فى إنشاء البرنامج وهو تداخل العبارات case بنفس الطريقة التى تتداخل بها العبارات الشرطية العادية . والبرنامج الآتى بعد شكل (٥ ـــ ١٣) يوضح كيفية برمجة الحالات الأربع الأولى وقد تركنا الحالات الأربع الثانية كتمرين .

البرنامج ٣

```
case over21 of
true :  case largecar of
        true :  case riskdriver of
                true : writeln(p45);
                false: writeln(p15)
                end;
        false:  case riskdriver of
                true : writeln(p30);
                false: writeln(ok)
                end
        end;
false:  تستكمل بقية الحالات بنفس الأسلوب
end
```

الحالات الأربع الأولى

شكل (٥ ـــ ١٣)

(٥ ـــ ٤) ملاحظات على بناء العبارات الشرطية المتداخلة :

رأينا أن العبارة الشرطية يمكن أن تحتضن بداخلها عدة عبارات شرطية أخرى بالصورة الآتية :

```
if...then
        if...then...
        else...
else
        if....the...
        else...
```

وترتيب العبارات بالصورة الموضحة لا يفيد الكومبيوتر فى شىء وإنما يفيد المبرمج فى ترتيب أفكاره وحتى يعرف « إلى أى if تنتمى كل else ؟ » ! والبرنامج بهذه الصورة سهل القراءة . لكن الكومبيوتر لن يمانع أن يستقبل البرنامج كالآتى :

if...then if...then...else...else if...then...else...

وحتى لا يخلط الكومبيوتر فى تبعية الكلمات else للكلمات if ، فإنه يعتبر أن أى else تتبع أقرب عبارة شرطية غير مكتملة . فهو كلما يصادف الكلمة else يبحث عن أقرب ثلاثى if – then – else (لاحظ الأسهم) .

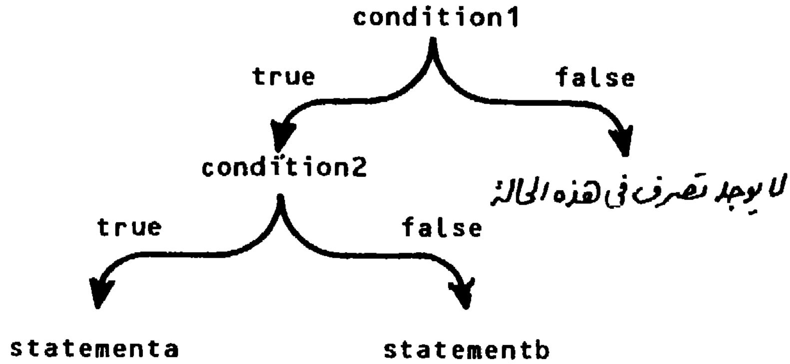
ولنعتبر هذا المثال :

```
if  condition1  then
    if  condition2  then  statementa
    else  statementb
```

شكل (٥ ـــ ١٤)

فى هذا المثال فإن الكلمة else تتبع العبارة الشرطية الثانية (الكلمة if الثانية) وبذلك فإن البرنامج سوف يتبع المنطق الموضح فى الشكل (٥ ـــ ١٥) فى تنفيذ

العبارات المختلفة وفقاً للشروط المختلفة .



شكل (٥ ــ ١٥)

فإذا أردنا تمثيل شروطاً حقيقية فلنعتبر هذه الحالة :

```
if x > 50 then
  if y > 50 then writeln('both values > 50')
  else writeln('only the first value is > 50')
```

شكل (٥ ــ ١٦)

فى هذا المثال سوف يقوم الكومبيوتر بطبع رسائل معينة عندما تكون $X>50$ (مع جميع قيم Y) أما الحالة التى تناظر $X<=50$ فلن يقابلها أى تصرّف .

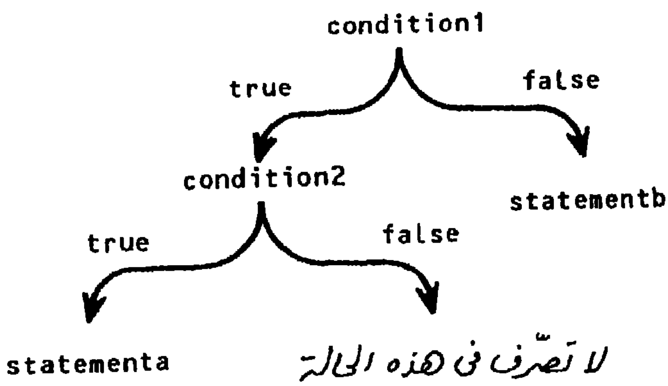
وبصفة عامة فيمكنك استخدام عبارات البداية والنهاية (begin, end) لتنظيم البرنامج الذى يحتوى على عبارات شرطية متداخلة . ففى البرنامج السابق

يمكننا تغيير تبعية else للعبارة الشرطية الثانية ونقل تبعيتها إلى العبارة الشرطية الأولى باستخدام begin,end كالآتى :

```
if  condition1  then
begin
   if  condition2  then  statementa
end
else statementb
```

شكل (٥ ١٦)

فى هذه الحالة سوف يتغير المنطق الذى يتبعه البرنامج ويصبح كما فى شكل (٥ ــ ١٧) .



شكل (٥ ــ ١٧)

فإذا عدنا للتمثيل بالأرقام نعتبر الصورة الجديدة للمثال :

```
if x > 50 then
begin
   if y > 50 then writeln('both values are > 50')
end
else writeln('the first value is not > 50')
```

شكل (٥ ــ ١٨)

فإذا تتبعنا منطق الصورة الجديدة للبرنامج سوف نرى أن الحالة التى لا يقابلها أى تصرف من الكومبيوتر هى الحالة التى تكون فيها قيمة X أكبر من **50** وقيمة Y أصغر أو تساوى **50** .

(٥ ــ ٥) مراجعة البيانات data validation :

عندما نبنى برنامجاً يحتوى على العديد من البدائل باستخدام العبارة case فإنه من المحتمل أن يحتوى متغيّر الاختيار قيمة مختلفة عن قيم العناوين الموجودة بالبرنامج . وفى هذه الحالة فإن البرنامج يتوقف مرسلاً رسالة بالخطأ الذى حدث .

ولحماية البرنامج ضد مثل هذه الأحوال فإن يلزم إضافة عبارة شرطية تخبر البرنامج بالتصرف المطلوب منه إذا حدث مثل هذا الخطأ .

وهذه الإضافة قد تكون بالصورة التالية :

```
if                                    then
 ┌──────────────────┐
 │  case-statement  │
 └──────────────────┘
else
```

شكل (٥ ــ ١٩)

ويلى الكلمة if الشرط المطلوب توفره بحيث يضمن احتواء متغير الاختيار على القيمة الصحيحة دائماً ، ويلى الكلمة else التصرف المطلوب من البرنامج فى حالة وجود قيمة مخالفة .

ولتوضيح استخدام هذا المنشأ نضرب مثالاً بماكينة العملة التى عالجناها من قبل فى الباب الرابع .

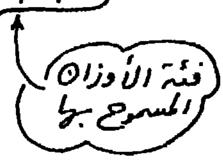
مثال (٥ ــ ٦) فئات العملات :

تستقبل ماكينة العملة قطعاً من العملة ذات أوزان مختلفة ٣٥ ، ١٦ ، ٩ ، ٧ وهذه الأوزان تناظر العملات ١٠ ، ٥ ، ١ ، $\frac{1}{2}$ قرشاً على الترتيب .. فإذا كانت العملة ذات وزن مختلف ، فعلى البرنامج أن يطرد العملة من الماكينة ويطبع الرسالة « COIN REJECTED » .

أما إذا كانت العملة هى أحد العملات المذكورة فيتم التعامل معها بالعبارة case كما سبق أن تعاملنا معها فى الباب الرابع

وشكل (٥ ــ ٢٠) يوضح الفقرة المطلوب إضافتها لمراجعة وزن العملات .

```
        if weight in [35,16,9,7] then
case weight of
35: amount:=amount+10;
16: amount:=amount+ 5;
9 : amount:=amount+ 1;
7 : amount:=amount+0.5
end
        else writeln('COIN REJECTED')
```



شكل (٥ ــ ٢٠)

وكما نرى أن التعبير الشرطى قد تضمن الفئة [35,16,9,7] المميزة بالقوسين المربعين . واستخدام الفئة (set) يغنى عن عبارات بوليانية كثيرة مثل :

```
(weight = 35) or (weight = 16) or (weight) = 9) or
(weight = 7 )
```

أما التعبير **weight in [35,16,9,7,]** فيعنى باختصار « اختبر ما إذا كانت قيمة المتغير weight هى أحد عناصر الفئة **[35,16,9,7]** .

واختبار الفئات يصلح لكثير من التطبيقات فعلى سبيل المثال يمكن اختبار قيمة المتغير y إذا ما كان يقع ما بين الواحد والعشرة (أى ينتمى إلى فئة الأعداد من واحد إلى عشرة) كالآتى :

y in [1..10]

والنقطتان المستخدمتان بداخل علامة الفئة تغنيان عن كتابة كل الأرقام بين الواحد والعشرة .

والتعبير السابق يكافئ تماماً التعبير المنطقى :

(y >= 1) and (y<=10)

والقاعدة المحددة لمثل هذه الفئة أنها لا يجوز أن تحوى على أعداد حقيقية (سوف نعرض الموضوع تفصيلاً فى الباب الثامن) .

ولنعد الآن لمثال ماكينة العملة لنراه فى صورته الكاملة ، وفى هذا المثال فإن ماكينة العملة تستقبل قطع النقود وتجمعها حتى إذا بلغت ١٢٣ قرشاً أو زادت عن ذلك فإن البرنامج يطبع الرسالة : قبلت العملات COINS ACCEPTED ويحسب مقدار الباقى من النقود إن وجد . والبرنامج موضح بشكل (٥ — ٢١) .

البرنامج

```
program coins(input,output);
var weight: integer;
    amount: real;
begin
    amount:=0;
    repeat
       read(weight);
       if weight in [35,16,9,7] then
case weight of
35: ambunt:=amount+10;
16: amount:=amount+ 5;
9 : amount:=amount+ 1;
7 : amount:=amount+0.5
end
       else writeln('COIN REJECTED')
    until amount >=123;

write('COINS ACCEPTED');
if amount > 123 then
writeln('CHANGE DUE: ',amount-123)
end.
```

ماكينة العملة

فئة الأوزان المسموح بها

رفض العملة

قبول العملات

حساب الباقي

شكل (٥ ــ ٢١)

(٥ ــ ٦) حماية البرنامج من البيانات الخاطئة :

فى بعض الأحيان تتسبب البيانات غير الصحيحة فى الحصول على نتائج غير صحيحة دون أن يتوقف البرنامج ودون أى رسالة تحذير من الكومبيوتر وهذه أخطر الحالات . لذلك يجب أن يتضمن البرنامج حماية ضد دخول مثل هذه البيانات . وطريقة الحماية تختلف بحسب نوعية البيانات الداخلة لكنه من

الشائع مع البيانات العددية على وجه الخصوص تحديد مدى معيّن للأرقام الداخلة للبرنامج فإذا خرجت عن هذا المدى أرسل البرنامج رسالة تحذير ! .

مثال (٥ ــ ٧) كم تكلفك الأجازة ؟

يقوم هذا البرنامج بحساب تكاليف أجازتك وكل ما يطلبه منك من بيانات هو اليوم والشهر الذى تبدأ فيه الأجازة واليوم والشهر الذى تنتهى فيه الأجازة . ويقوم البرنامج بعد ذلك بحساب مدة الأجازة باستخدام هذه التواريخ وحساب تكلفة كل يوم باستخدام ٣ معدلات ثابتة فى البرنامج هى ٥٧ جنيهاً ، ٧٢ جنيهاً ، ٨٧ جنيهاً . والبرنامج يختار بين هذه المعدلات بحسب شهور السنة فشهور الصيف ٦ ، ٧ ، ٨ ، ٩ من الشهور المكلفة أما شهور الشتاء ١ ، ٢، ١١ ، ١٢.فهى شهور منخفضة التكاليف أما أشهر الربيع والخريف فهى معتدلة التكاليف . وبالطبع يمكنك تغيير هذه المعدلات بحسب المكان الذى تقضى فيه أجازتك .

تكاليف الاجازة

```
program holiday(input,output) ;
const lowrate = 57;
      midrate = 72;
      peakrate= 87;

var day1, month1, day2, month2, duration,
    tillendofmonth, cost  :  integer;

begin

 read(day1, month1, day2, month2) ;

 if (day1 in [1..31] )
      and (day2 in [1..31])
        and (month1 in [1..12])
            and (month2 in [1..12])
              and (month1 <= month2)
     then
```

اختبار البيانات

```
begin
     if month1 = month2   then
        duration := day2 - day1  +1
     else
     begin
       case month1 of
        2  :tillendofmonth  := 28 - day1;
        9,4,6,11 :tillendofmonth := 30 - day1;
        1,3,5,7,
        8,10,12  :tillendofmonth := 31 - day1
       end;
         duration:= day2 + tillendofmonth + 1
        end;

        case month2 of
        1,2,11,12   :cost   := duration*lowrate;
        3,4,5,10    :cost   := duration*midrate;
        6,7,8,9     :cost   := duration*peakrate
        end;

        write (' cost of holiday is ', cost )
end
```

معالجة البيانات

```
else write ('you have typed erroneous dates')

end.
```

رسالة خطأ عند إدخال بيانات غير صحيحة

شكل (٥ ــ ٢٢)

أما الاختبار المطلوب إجراؤه على البيانات الداخلة للبرنامج فهو التأكد من أن الأرقام الممثلة للأيام تقع فى الفئة [1..31] والأرقام الممثلة للشهور تقع فى الفئة [1..12] وأن الشهر الذى تبدأ فيه الإجازة يساوى أو أكبر من

الشهر الذى تنتهى فيه الأجازة .

أما متغيرات البرنامج المستخدمة فهى كالآتى :

(١) الثوابت :

lowrate	معدّل تكلفة يوم أجازة فى الشتاء
midrate	معدّل تكلفة يوم أجازة فى الربيع أو الخريف
peakrate	معدّل تكلفة يوم أجازة فى الصيف

(٢) المتغيرات :

day1	يوم نهاية الأجازة
day2	يوم نهاية الأجازة
month1	شهر بداية الأجازة
month2	شهر نهاية الأجازة
duration	مدة الأجازة
tillendofmonth	المدة حتى نهاية الشهر
cost	تكاليف الرحلة

إضافة أخرى :

فى البرنامج السابق لا يتم معالجة البيانات قبل اختبارها وبالتالى فإن البيانات الصحيحة فقط هى التى تمر إلى الجزء من البرنامج المحصور داخل المربع . أما البيانات غير الصحيحة فينتج عنها توقف البرنامج وإرسال رسالة الخطأ المذكورة .

ولكن البرنامج الجيد هو الذى لا يتوقف لأى سبب من الأسباب بل دائماً يمنح فرصة أخرى لتصحيح الخطأ .

ويمكن إضافة الجزء التالى إلى البرنامج السابق حتى يكتسب هذه الخاصية ، حيث ينبّه إلى الخطأ الذى وقع ويطلب منك إعادة إدخال البيانات . شكل (٥ ــ ٢٣) .

إضافة لبرنامج تكاليف الإجازة

```
read(day1,month1,day2,month2);

while not ((day1 in [1..31]) and
           (day2 in [1..31]) and
           (month1 in [1..12]) and
           (month2 in [1..12]) and
           (month1 <= month2))     do

begin
 write('you''ve made an error please retype');
 read(day1,month1,day2,month2)
 end;
```

شكل (٥ ــ ٢٣)



■ مسائل على الباب الخامس :

س (٥ ـــ ١) تقوم إحدى الشركات بتحليل يومى للمبيعات التى تمت خلال يوم كامل ويتضمن التحليل طبع قائمة بعدد السلع المباعة والتى يتعدى سعرها ١٠ جنيهات . ويتم هذا الحصر بالنسبة لجميع أقسام الشركة التى يبلغ عددها ٦٣ قسماً . أكتب البرنامج الذى يقوم بهذا العمل علماً بأن المدخلات لهذا البرنامج عبارة عن قائمة بأسعار الأصناف المباعة فى اليوم المعين ، وتنتهى قائمة الأسعار برقم سالب مثل (1−) .

س(٥ ـــ ٢) فى مثال المراقبة الرقمية للأنتاج تم تصنيف أطوال المنتجات إلى نوعين :

- أطوال تقع فى النطاق المقبول ، وهى التى لا يزيد الفرق بينها وبين الطول القياسى (6.37) عن 0.1 .
- أطوال تقع خارج النطاق المقبول وهى الأطوال التى تتعدى الفرق السابق زيادة أو نقصاً .

والمطلوب تطوير البرنامج لكى يقوم بتصنيف الأطوال غير المقبولة إلى أطوال « أكبر من اللازم » وأطوال « أصغر من اللازم » .

س (٥ ـــ ٣) أكتب برنامجاً يستقبل رقمى شهرين من نفس العام متبوعين برقم السنة ، ويطبع عدد الأيام من بداية الشهر الأول وحتى نهاية الشهر الثانى .

المطلوب تطوير البرنامج السابق ليستقبل مجموعتين من الأرقام معبرة عن التاريخ (اليوم والشهر) بالصورة الآتية :

```
21  2
15  7
```

(بمعنى ٢١ فبراير و١٥ يوليو)

ويلى هذه الأرقام الرقم الدال على السنة . والمطلوب حساب الفترة التى تقع بين هذين التاريخين بالأيام وطباعتها على الشاشة .



الباب السادس

التعامل مع اللبنات
(CHARACTER Manipulation)

مفتتح

حتى الآن ... قد تعاملنا مع الأعداد والبيانات العدديّة كمدخلات ومخرجات ومادة للمعالجة . وفى هذا الباب سوف نلتقى بالحرفيّات أو البيانات الحرفيّة التى تمثّل أسماء الأشخاص والعناوين وأرقام التليفونات . وقد علمنا من قبل أن لغة باسكال تتعامل مع الحرفيات كلبنات مفردة (characters) .

فلغة باسكال القياسية ترى الحرفيات سواء كانت أسماء أو أرقام تليفونات أو نصوصاً أدبية ، كمجموعة من اللبنات المتتالية . ومع ذلك فلنا عودة أخرى لموضوع الحرفيات بمعناها المعروف فى الباب الثامن حيث نعرض الإمكانات التى قدمتها الطرازات الجديدة للغة لخدمة معالجة الكلمات والنصوص .

(٦ ــ ١) متغيّرات اللبنات *(character variables)* :

علمنا من قبل أن اللبنات تنتمى إلى نمط خاص من الأنماط القياسية فى اللغة والمسمى بنمط اللبنة (CHAR) . ومتغيّر اللبنة لا يحتوى إلاّ على لبنة واحدة فقط ، قد تكون حرفاً أو رقماً أو علامة خاصة من العلامات التى على لوحة الأزرار .

وفى الأجزاء التالية سوف نعرف بعض الأمثلة لمعالجة اللبنات بالطرق المختلفة .

مثال (٦ ــ ١) :

يقرأ هذا البرنامج ثلاث لبنات متتابعة ويطبعها فى ترتيب معكوس . وقد أطلق الاسم "first" على المتغيّر الأول والاسم "second" على الثانى ، والاسم "third" على الثالث .

```
program reverse(input,output);

var first, second, third : char;

begin
   read(first, second, third);
   writeln(third, second, first)
end.
```

شكل (٦ ــ ١)

عند تشغيل البرنامج سوف يتوقف عند العبارة read منتظراً إدخال اللبنات الثلاث . فإذا أدخلنا له ثلاثة أحرف مثل :

bat

فإنه يحفظها فى الذاكرة فى ثلاثة أوعية مستقلة يحمل كل منها اسم المتغير الخاص بها كما فى الشكل (٦ — ٢) .

first 'b'   second 'a'   third 't'

شكل (٦ — ٢)

لذلك فعند تنفيذ أمر الطباعة سوف يسترجع البرنامج الحروف الثلاثة من الذاكرة بحسب ترتيب المتغيرات فى عبارة الطباعة (وهو ترتيب معكوس) ويبدأ بطباعة المتغير الثالث ثم الثانى ثم الأول فتكون النتيجة هى الحروف :

tab

وبالطبع يجوز إدخال ثلاث علامات خاصة مثل :

* + -

فيصبح الخرج : - + *

أو إدخال ثلاثة أعداد مثل :

123

فيصبح الخرج : **321**

ولا يفوتنا ملاحظة أن هذه الأرقام لا يمكن إجراء عليها أية عمليات حسابية ، فهى ليست أكثر من لبنات متتابعة .

مثال (٦ — ٢) :

يقرأ هذا البرنامج ثلاثة لبنات ويطبعها بكل التباديل الممكنة . وللبرنامج خاصية هامة عند قراءة اللبنات فهو يتجاهل أية مسافات خالية قد تكون موجودة قبل أول لبنة من اللبنات الثلاث .

```
program anagrams(input,output);

var first, second, third : char;

begin
   repeat
      read(first)
   until first <> ' ';

   read(second, third);

   writeln(first,second,third, '  ', first,third,second);
   writeln(second,first,third, '  ', second,third,first);
   writeln(third,first,second, '  ', third,second,first)
end.
```

قراءة اللبنة الأولى مع تجاهل المسافات الخالية

قراءة اللبنة الثانية والثالثة

شكل (٦ — ٣)

وعند تنفيذ هذا البرنامج سوف ندخل له ثلاث لبنات متتاعبة ولتكن : cat (يجوز ترك أية مسافات خالية قبل أول لبنة) . عندئذ يقوم البرنامج بطباعة التباديل الآتية :

```
cat  cta
act  atc
tca  tac
```

(٦ — ٢) نهاية السطر eol (end-of-line) :

عندما ندخل بياناً من البيانات فإننا نضغط على الزر (ENTER) أو الزر (RETURN) فى نهاية الإدخال لإعلام الكومبيوتر بأن البيان المطلوب جاهز على الإدخال والمعالجة . وما يحدث فى حقيقة الأمر أن الضغط على زر إدخال البيانات يضيف إلى البيان المكتوب لبنة جديدة تسمى علامة نهاية السطر (End-of-line marker) . وحتى يمكن إدخال البيانات سطراً بسطر فإن لغة باسكال تمدنا بالدالة eol (وهى اختصار العبارة end-of-line) ، وهذه الدالة من النوع المنطقى (أو البوليانى) أى أنها تحتوى إما على القيمة true أو القيمة false . وهى تستخدم لاختبار وجود علامة نهاية السطر فى البيان المُدخل عند قراءة كل لبنة تالية .

والبرنامج التالى يستقبل مجموعة من اللبنات ويقوم بحساب عدد اللبنات التى تمت كتابتها قبل الضغط على زر نهاية السطر (الزر ENTER أو RETURN) :

```
var nextchar : char;  count : integer;
    .
    .
    count := 0;
    while not eoln(input) do
    begin
       read(nextchar);
       count := count + 1
    end
```

شكل (٦ — ٤)

* **ملاحظة** : قد لا يعمل هذا البرنامج مع بعض الطرازات) .

والدالة **eol** قد تكتب بالصورة **eol(input)** أو بالصورة المبسطة **eol** مع بعض الطرازات .

وعادة ما يتعامل برنامج باسكال مع علامة نهاية السطر كمسافة خالية وحتى نشاهد ذلك يمكن تجربة هذا البرنامج :

```
var ch1, ch2, ch3, ch4 : char;
    .
    .
    .
    read(ch1, ch2, ch3, ch4);
    write(ch1, ch2, ch3, ch4)
```

شكل (٦ — ٥)

فى هذا البرنامج سوف يتوقف التنفيذ عند العبارة read منتظر إدخال أربع لبنات متتابعة ثم الضغط على زر نهاية السطر . فإذا كتبنا الحروف الآتية كمُدخل :

abcd

فإن البرنامج سيطبع نفس الحروف بموجب العبارة writeln :

abcd

لنجرب الآن إدخال الحروف الأربعة بطريقة أخرى فنكتب الحرفين ab ثم نضغط زر نهاية السطر ثم نضغط على الحرف C ثم زر نهاية السطر ، فى هذه الحالة نحصل على الجواب :

ab c

أى أن البرنامج قد اعتبر أن علامة نهاية السطر (الممثلة بالمسافة الخالية) هى أحد المدخلات الأربعة .

ولنجرب أيضاً إدخال الحروف الأربعة abcd مع الضغط على الزر enter

(زر نهاية السطر) بعد كل حرف . عندئذ سنحصل على الجواب مباشرة بعد الضغطة الثانية قبل أن نكمل إدخال الحروف كلها وسيكون الجواب كالآتى :

a b

فاللبنات الأربع المدخلة فى هذه الحالة كانت :

- الحرف a .
- علامة نهاية السطر .
- الحرف b .
- علامة نهاية السطر .

ولو أننا كتبنا مجموعة كبيرة من الحروف بعد abcd سيظل الجواب دائماً هو abcd فعبارة القراءة فى هذا البرنامج لا تتوقع إلاّ أربعة حروف فقط !

وجرب هذا المثال :

المدخلات :	**abcdefghijk**
النتيجة :	**abcd**

(٦ ــ ٣) *نهاية الملف (eof) end-of-file :*

تعتبر لغة باسكال أن كل المدخلات (input) والمخرجات (output) تأتى من ملفات (files) وتذهب إلى الملفات[١] . وقد تكون الملفات هى ملفات حقة بمعناها المتعارف عليه فى لغات الكومبيوتر المختلفة بمعنى أنها مجموعة مترابطة من سجلّات البيانات[١] (records) التى تحفظ على القرص المغنطيسى أو الشريط المغنطيسى ولكنها أيضاً قد تكون مجموعة من اللبنات المدخلة من لوحة الازرار أو الخارجة إلى الشاشة أو جهاز الطباعة .

لذلك فإن هناك نوعين من الملفات يتم فتحهما آلياً ببرنامج باسكال **هما ملف**

(١) للمزيد من التفصيلات عن الملفات يرجع لكتابنا « مدخلك إلى عالم الكومبيوتر » .

الدخل input ويستخدم لإدخال البيانات إلى البرنامج من جهاز الدخل القياسى (لوحة الأزرار فى حالة الميكروكومبيوتر) ، **وملف الخرج output** وهو يستخدم لإخراج المعلومات إلى جهاز الخرج القياسى (الشاشة أو جهاز الطباعة) .

ولا حاجة بك لفتح هذه الملفات إذا أردت استخدامها ، أما الملفات الأخرى التى قد تكون فى وسط آخر من أوساط الملفات مثل القرص المغنطيسى فتحتاج إلى إعلان مسبق .

ولسنا بصدد التعامل مع الملفات كموضوع مستقل الآن ، بل سنكتفى بالتعرض لملفات أجهزة الدخل والخرج القياسية وهى مجموعة اللبنات التى ترسل من لوحة الأزرار إلى البرنامج .

والدالة الجديدة التى تمدنا بها لغة باسكال هى دالة نهاية الملف **(eof)** والتى تستخدم لتعريف البرنامج بأن البيانات المدخلة قد بلغت نهايتها .

ودالة نهاية الملف قد تكتب بالصورة **eof** أو بالصورة الكاملة **eof(input)** .

والبرنامج التالى يقوم بحساب عدد اللبنات المدخلة بما فى ذلك لبنة نهاية السطر **(eol)** باستخدام دالة نهاية الملف .

```
count := 0;
repeat
   read(nextchar);  count := count + 1
until eof(input);
writeln('there are ', count, ' characters in the file.
```

(تعليقات مكتوبة بخط اليد: كرر العمل حتى تبلغ نهاية الملف — عدد اللبنات)

شكل (٦ ـــ ٦)

* **ملاحظة** : دالة نهاية الملف قد يختلف استخدامها مع الطرازات المختلفة (يرجع لكتاب اللغة للجهاز) .

مثال (٦ ــ ٣) كم مرة يتكرر الحرف فى النص ؟

من التطبيقات المفيدة فى معالجة اللبنات حساب عدد مرّات تكرار حرف معيّن فى سطر ما أو نص ما .

والبرنامج التالى يقرأ النص حرفاً حرفاً ويضعه فى المتغير nextch ثم يقارنه بالحرف e المخزن فى الثابت المسمى ginench فإذا حدث التطابق أضاف واحداً إلى العدّاد noff occurrences وفى النهاية عندما يصل البرنامج إلى نهاية الملف يطبع محتوى العدّاد الذى يمثل عدد مرات التكرار .

```
program charcount(input,output);

const givench = 'e';

var noofoccurrences : integer;  nextch : char;

begin
   noofoccurrences := 0;
   repeat
      read(nextch);
      if nextch = givench then
         noofoccurrences := noofoccurrences + 1
   until eof(input);
   writeln('the character ', givench, ' appears ',
         noofoccurrences, ' times.')
end.
```

شكل (٦ ــ ٧)

(٦ ــ ٤) *ترتيب اللبنات :*

تأخذ اللبنات ترتيباً معيناً فى تعاقبها ، وتمنح لغة باسكال رقماً لكل لبنة يمثل ترتيب هذه اللبنة فى التتابع فاللبنة الأولى تأخذ الرقم صفراً **(0)** والثابتة تأخذ الرقم 1 والثالثة 2 وهكذا ...

وتتابع الحروف الأبجدية من A إلى Z تتابعـــاً متصلاً لا يتخلله فواصل أى أن الحرف A يأتى قبل B والحرف B يأتى قبل C وهكذا ... كما تتابع الأرقام أيضاً تتابعاً متصلاً . أما العلامات الخاصة فقد يختلف ترتيبها من طراز لآخر فى لغة باسكال .

وكما نستخدم المؤثرات العلاقية مع الأعداد مثل :

x>y,z<=2

يمكننا استخدام المؤثرات العلاقية أيضاً مع اللبنات فنقول :

'a'<'b'

وإذا كان ch متغير لبنة فيجوز أن نكتب :

ch<='a'

أو **ch>='z'**

وبذلك يمكن اختبار محتوى متغير اللبنة لمعرفة ما إذا كان يحتوى على حرف من الحروف باستخدام التعبير العلاق :

(ch>='a') and (ch<='z')

وباستخدام الفئات (sets) مع المؤثر **in** يمكن التعبير عن نفس المضمون بطريقة إفضل :

ch in ['a'..'z']

كذلك لمعرفة ما إذا كان المتغير ch يحتوى على رقم يمكن استخدام التعبير العلاق :

(ch>='0') and (ch<='9')

أو باستخدام الفئات أيضاً :

ch in ['0'..'9']

مثال (٦ ـــ ٤) تصنيف اللبنات :

يقوم هذا البرنامج باستقبال لبنة واحدة ثم يطبع رسالة على الشاشة ليخبرنا بنوع اللبنة هل هى حرف (alphabetic) أو رقم (numeric) أو ما عدا ذلك (not alphanumeric) .

والاختبار يتم على كل من فئة الحروف **['a'..'z']** وفئة الأرقام **['o'..'g']** فإذا كانت اللبنة لا تنتمى لإحدى الفئتين كانت لبنة خاصة (not alphanumeric) . شكل (٦ ـــ ٨) .

```
program classifychar(input,output);

var character : char;

begin
   read(character);
   if character in ['a'..'z'] then
      writeln('that character is alphabetic.')
   else if character in ['0'..'9'] then
           writeln('that character is numeric.')
        else
       writeln('that character is not alphanumeric.')
end.
```

تصنيف اللبنات

فئة الحروف

فئة الأرقام

شكل (٦ ـــ ٨)

مثال (٦ ـــ ٥) كم حرفاً فى الكلمة ؟

يستقبل هذا البرنامج كلمة ما ويحسب عدد حروفها .. وقد تضمّن البرنامج شرطاً لتجاهل أى لبنة غير أبجدية (ليست حرفاً) قد تقع فى بداية الكلمة ،

وكذلك فإن الكلمة المدخلة يعقبها لبنة غير أبجدية كعلامة لنهاية الإدخال .

(**ملاحظة** : سوف نعرض فى الباب السابع دالة جاهزة لحساب طول الكلمة مع موضوع الحرفيات (strings)) .

```
program wordlength(input,output);

var nextchar : char;  noofletters : integer;

begin                                    حساب طول الكلمة
   repeat
      read(nextchar)
   until nextchar in ['a'..'z']:
  حتى هذه النقطة سوف يحتوى المتغير nextchar على الحرف الأول من الكلمة
   noofletters := 0;
   repeat
      noofletters := noofletters + 1;
      read(nextchar)
   until not (nextchar in ['a'..'z']);
      حتى هذه النقطة سوف يحتوى المتغير nextchar على اللبنة التالية للكلمة

   writeln('no. of letters: ', noofletters)
end.
```

شكل (٦ ــ ٩)

مثال (٦ ــ ٦) النسبة المئوية لتكرار حرف :

أما البرنامج التالى هو يستقبل قطعة من نص ويقوم بحساب نسبة تكرار حرف معيّن فى هذا النص . والحرف المطلوب حساب معدّل تكراره فى هذا المثال هو الحرف "e" .

والخرج المتوقع من البرنامج يكون على الصورة :

16% of letters were e-s

والحرف s الذى يعقب الحرف e لزوم الجمع فى اللغة الانجليزية !

```
program letterfrequ(input,output);

const givenletter = 'e';

var noofletters, noofgivenletter : integer;
    character : char;

begin
   noofletters := 0;  noofgivenletter := 0;
   repeat
      read(character);
      if character in ['a'..'z'] then
      begin
         noofletters := noofletters + 1;
         if character = givenletter then
            noofgivenletter := noofgivenletter + 1
      end
   until eof(input);
   writeln(noofgivenletter/noofletters*100,
        '% of the letters were ', givenletter, '-s.')
end.
```

شكل (٦ ــ ١٠)

ويترك للقارىء محاولة إنشاء برنامج لحساب عدد الكلمات فى النص كتدريب .

(٦ ــ ٥) المزيد من الوسائل لمعالجة اللبنات :

(٦ ــ ٥ ــ ١) طباعة اللبنات على الشاشة :

يمكنك أن ترى الحروف الأبجدية مرتبة على الشاشة باستخدام هذه الشريحة الصغيرة :

البرنامج

```
var letter : char;
    .
    .
    .
    for letter := 'a' to 'z' do
       write(letter)
```

التنفيذ

```
abcdefghijklmnopqrstuvwxyz
```

شكل (٦ ــ ١١)

وإذا استبدلنا عبارة التكرار « التصاعدية » بعبارة التكرار « التنازلية » downto فإن الحروف تطبع من الآخر إلى الأول أى من z إلى a .

البرنامج

```
    .
    .
    .
    for letter := 'z' downto 'a' do
       write(letter)
    .
    .
    .
```

التنفيذ

```
zyxwvutsrqponmlkjihgfedcba
```

شكل (٦ ــ ١٢)

(٦ ــ ٥ ــ ٢) دوال اللبنات :

وفى الأجزاء التالية سوف نتعرض بتفصيل أكثر لخصائص اللبنات وترتيبها وطرق معالجتها بالدوال المختلفة .

• الدالة **pred** : تستخدم هذه الدالة لإيجاد اللبنة السابقة للبنة أخرى فى الترتيب ؛ وهى اختصار كلمة predecessor . فالعبارة :

writeln(pred('b'))

سوف تطبع الحرف a وهو الحرف السابق للحرف b المستخدم كدليل للدالة pred

• الدالة **succ** : وهى الدالة المقابلة للدالة pred وهى مختصر كلمة successor ، وتستخدم لإيجاد اللبنة التالية للبنة أخرى . فالعبارة :

writeln (succ('b'))

سوف تطبع الحرف **c** التالى للحرف **b** .

مثال (٦ ــ ٧) رسالة مشفّرة :

يقوم أحد رجال المخابرات بإرسال رسالة سرية إلى الكومبيوتر الرئيسى بمركز القيادة وحتى تتحقق عناصر السرّية الكاملة فإنه يقوم بتشفير الرسالة باستخدام البرنامج الآتى الذى يستبدل كل حرف من الرسالة بالحرف التالى له فى الأبجدية كما يستبدل الحرف z بالحرف a . يقرأ البرنامج الرسالة الأصلية من ملف ثم يطبعها فى صورتها المشفرة .

رسالة مشفرة

```
program code(input,output);

var character : char;

begin

   repeat

      while not eoln(input) do
      begin
         read(character);
         if character in ['a'..'z'] then
            if character = 'z' then write('a')
            else write(succ(character))
         else write(character);
      end;

      readln;  writeln
   until eof(input)

end.
```

شكل (٦ — ١٣)

نلاحظ فى هذا البرنامج ظهور عبارة قراءة جديدة هى readln وهى تستخدم لبدء سطر جديد من سطور **الدخل** ، تماماً كما تستخدم العبارة writeln لبدء سطر جديد من سطور الخرج .

*** الدالتان chr و ord :**

ويسمى هذا العدد لترتيب اللبنة بالعدد الترتيبى (ordinal number) وتفيدنا الدالة (ord) فى تحويل اللبنة إلى العدد الترتيبى المناظر لها . أما الدالة (chr) فهى تحول العدد الترتيبى إلى اللبنة المناظرة له .

أى أن :

ord (char)

تعطى هذه الدالة العدد الترتيبى المناظرة للبنة (char) . ودليل هذه الدالة يكون دائما لبنة أو متغير لبنة .

chr (integer)

تعطى هذه الدالة اللبنة المناظرة للعدد الصحيح الذى تؤثر عليه (integer) . والدليل لهذه الدالة يكون دائماً عدداً صحيحاً أو متغيّراً صحيحاً .

وترتيب اللبنات قد يختلف من طراز إلى آخر من طرازات لغة باسكال .

مثال (٦ ــ ٨) :

البرنامج التالى يطبع اللبنات من صفر إلى ٦٣ وأمام كل منها العدد الترتيبى الخاص بها (ordinal number) .

```
var ordinal : integer;
  .
  .
  for ordinal := 0 to 63 do
      writeln('character ', chr(ordinal),
               ' has ordinal number ', ordinal)
```

شكل (٦ ــ ١٤)

أما شريحة البرنامج التالية فهى تطبع العدد الترتيبى لكل لبنة من اللبنات الأبجدية .

```
var ch : char;
   .
   .
   for ch := 'a' to 'z' do
      writeln(ch, '  ', ord(ch))
```

شكل (٦ ــ ١٥)

أما شكل (٦ ــ ١٦) فيوضح البرنامج والتنفيذ على الكومبيوتر **IBM** ونرى أن التنفيذ يبدأ من الحرف m حتى الحرف z (بقدر ما تتسع الشاشة) ولعلنا نلاحظ أن ترتيب الحروف هو نفسه الكودآسكى (ASCII) . حيث يبدأ الحرف a عند 97 وينتهى الحرف z عند 122

(بدءاً من الحرف m)

```
var ch:char;
begin
for ch:='m' to 'z' do
writeln(ch,' ',ord(ch));
end.
```

```
Running
m 109
n 110
o 111
p 112
q 113
r 114
s 115
t 116
u 117
v 118
w 119
x 120
y 121
z 122

>
```

تنفيذ البرنامج
على الكومبيوتر IBM

شكل (٦ ــ ١٦)

ويمكن طباعة الحروف متجاورة باستخدام عبارة الطباعة write بدلاً من writeln كما فى الشكل الآتى .

وفى هذا البرنامج استبدلنا الحروف الصغيرة بالحروف الكبيرة من A إلى Z (من 65 إلى **90**) .

البرنامج

لاحظ عبارة الطباعة

```
var ch:char;
begin
for ch:='A' to 'Z' do
write(ch,' ',ord(ch),'  ');
end.
```

التنفيذ

```
>
Running
A 65  B 66  C 67  D 68  E 69  F 70  G 71  H 72  I 73  J 74  K 75  L 76  M 77  N
78  O 79  P 80  Q 81  R 82  S 83  T 84  U 85  V 86  W 87  X 88  Y 89  Z 90
>
```

شكل (٦ ــ ١٧)

ولا أهمية كبيرة لمعرفة الأعداد الترتيبية بل يكفى معرفة الترتيب النسبى للحروف الأبجدية . والبرنامج التالى يطبع لك **الحرف النونى** (الحرف ذو الترتيب ن) من الأبجدية عندما تدخل له قيمة المتغير n الذى يمثل ترتيب الحرف بالنسبة للحرف "a" إذا اعتبرنا أن الحرف a ترتيبه « الأول » .

إيجاد الترتيب النسبى للحرف

```
var n:integer;
begin
readln(n);
writeln('letter  ',n,' is ',chr(ord('a')+n-1));
end.
```

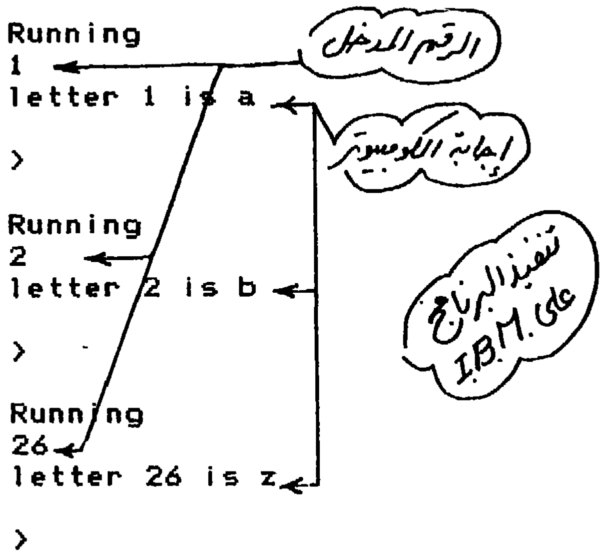
الترتيب

الحرف أو اللبنة

شكل (٦ ــ ١٨)

ونلاحظ فى هذا المثال ظهور عبارة القراءة readln وفائدتها هنا فصل الخرج (إجابة الكومبيوتر) عن الدخل (البيان المدخل) حتى لا يظهران متجاورين على نفس السطر . ويمكن أداء نفس الغرض باستخدام read يليها writeln لترك سطر خال .

والآتى بعد تنفيذ البرنامج على الكومبيوتر آى . بى . إم (IBM) حيث ندخل له رقماً من الأرقام فيخبرنا عن الحرف المناظر لهذا الرقم والممثل لترتيبه النسبى بالنسبة للحرف a .



شكل (٦ ــ ١٩)

■ **تمرينات على الباب السادس :**

س (٦ — ١) اكتب برنامجاً لتشفير رسالة سرّية بحيث يستبدل كل حرف من حروف الرسالة بالحرف الذى يليه بخمسة حروف فى الترتيب الأبجدى ، مع معاملة الحروف الأبجدية كنظام دائرى بمعنى أن الحرف a هو الحرف التالى للحرف z .

س (٦ — ٢) اكتب برنامجاً لقراءة ملف يحتوى على نص ويطبع عدد الحروف الأبجدية الموجودة بالنص كله كما يطبع رسالة أخرى لبيان عدد اللبنات غير الأبجدية بما فى ذلك علامة نهاية السطر .

س (٦ — ٣) اكتب برنامجاً يحسب ويطبع عدد الكلمات فى جملة . ولنفرض أن الجملة مميّزة بنقطة فى نهايتها . احسب أيضاً عدد الكلمات المكونة من أربعة حروف واطبع عددها .

يمكنك الاستعانة بالمنطق الآتى :

● كرّر الآتى :

— ابحث عن وحدّد بداية الكلمة التالية .

— اقرأ الكلمة التالية واحسب عدد حروفها .

— زد عدد الكلمات بمقدار واحد .

— إذا كان عدد حروف الكلمة = ٤ فزد عدد الكلمات الرباعية بمقدار واحد .

● حتى تصل إلى النقطة '.' .

س (٦ — ٤) اكتب برنامجاً يقرأ جزءاً من أحد النصوص ويختبر مدى سلامة النص من جهة القاعدة الإملائية الآتية :

« فى أى كلمة يأتى الحرف (i) قبل الحرف (e) ما لم يسبقه الحرف (c) » .

الباب السابع

المصفوفات

Arrays

مفتــح

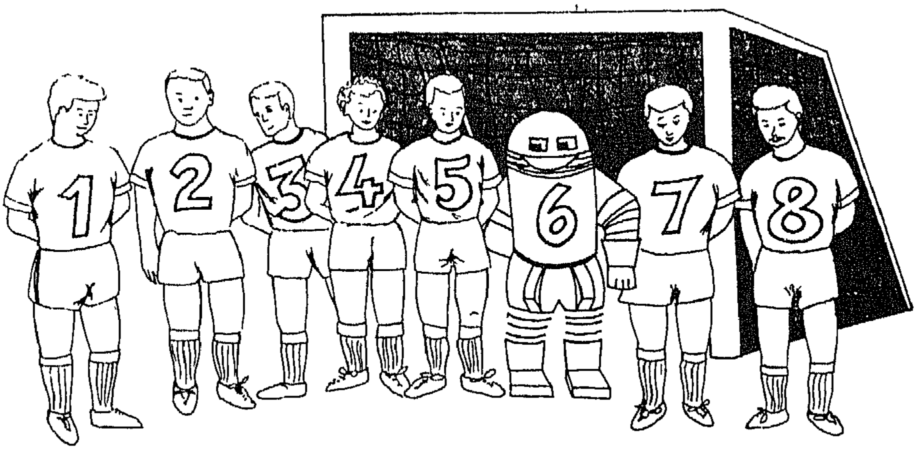
هل شاهدت مباراة لكرة القدم تلعب فيها فرق أجنبية ؟ وهل لاحظت أن المعلّق الرياضى غالباً ما يستخدم الأرقام عندما يتحدث عن اللاعبين الأجانب ؟ فيقول اللاعب رقم 2 واللاعب رقم 4 ؟ لماذا يستعين بالأرقام ؟

لأنها وسيلة أسهل لتمييز اللاعبين خاصة إذا كانت أسماؤهم جديدة علينا .

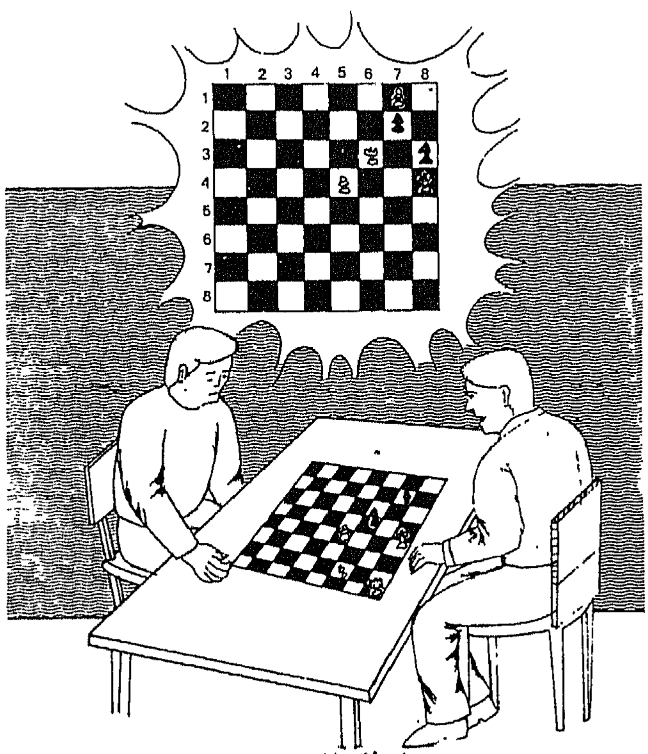
ومن وجهة نظر كومبيوترية : فإننا نقول أن لاعبى كرة القدم يكوّنون مصفوفة واحدة . وكل لاعب منهم يعتبر عنصراً من عناصر المصفوفة يميز برقمه بالنسبة لباقى عناصر مصفوفة الفريق ، وهذا أسهل بكثير من تذكر أسماء اللاعبين .

كذلك فإذا أردت أن تصف مباراة للشطرنج فلا أعتقد أن هناك بديلاً عن استخدام الأرقام تمييز مربعات لوحة الشطرنج . وفى هذه الحالة سوف يلزمك رقمان تمييز كل مربع : رقم يدل على الصف (الأفقى) وآخر يدل على العمود (الرأسى) . فتقول مثلاً أن الوزير الأبيض موجود بالمربع رقم (1،4) ، وهذا هو الأسلوب المتبع فعلاً فى لعب الشطرنج على الكومبيوتر حيث يسألك البرنامج عن رقم المربع الذى تريد الانتقال إليه . والشطرنج ينتمى إلى نوعية أخرى من المصفوّفة تسمى المصفوفة ذات البعدين .

والبديل الوحيد للتعبير عن مربعات الشطرنج بدون استخدام أسلوب المصفوفات هو أن نبتكر إسماً لكل مربع ويا له من عمل مزعج !



مصفوفة ذات بعد واحد



مصفوفة ذات بعدين

شكل (٧ ـــ ١)

(٧ ــ ١) المتــغيرات الدليليــــة (subscripted : variables)

تظهر الحاجة إلى استخدام المصفوفات عند التعامل مع البيانات المجدولة مثل درجات تلميذ فى مختلف المواد الدراسية أو درجات فصل دراسى بأكمله .

وتعرّف المصفوفة عموماً بأنها :

« مجموعة مفهرسة من البيانات »

فإذا أردنا التعبير عن درجات أحد الطلبة فى المواد الدراسية المختلفة فإنه من الممكن أن نعطى كل مادة اسم متغير مستقل مثل :

ARABIC , ENGLISH, MAT,....

ولكن من الأفضل أن نختار اسماً عاماً للمواد الدراسية وليكن :

SUBJECT [I]

حيث يسمى المتغير I بالدليل وهو يأخذ قيماً مختلفة ، ويمكن فى هذه الحالة أن تمثل درجات المواد المختلفة كالآتى :

SUBJECT [1], SUBJECT[2] , SUBJECT [3],..

وتسمى هذه النوعية من المتغيرات **بالمتغيرات الدليلة لأنها تعتمد على ذليل عام هو I** فإذا أخذ الدليل القيمة 1 فذلك يدل على درجة اللغة العربية ARABIC وإذا أخذ القيمة 2 دل المتغير على درجة الانجليزية ENGLISH وهكذا وهذا يغنى عن اختيار أسماء متغيّرات كثيرة .

وتشكل جميع عناصر المتغير الدليلى SUBJECT[1] مصفوفة ذات بعدواحد **(one dimensional array)** .

وقد مررنا بهذه النوعية من البيانات فى الأجزاء السابقة وقد أطلقنا عليها اسم الفئة (set) مثل فئة الحروف الأبجدية **['A'..'Z']** وفئة الأرقام **[0..9]** والحقيقة أنه لا فرق بين الاسمين فالحروف الأبجدية هى مصفوفة ذات بعد واحد وهى عناصر فئة الحروف الأبجدية . كذلك فإن لاعبى كرة القدم ينتمون جميعاً لفئة واحدة هى الفريق .

فإذا أتينا إلى حالة تمثيل نتائج مجموعة كبيرة من الطلبة فإننا عادة ما نكتب النتيجة فى صورة جدول كالموضح فى شكل (٧ ــ ٢) الذى نطلق عليه **المصفوفة ذات البعدين[١] (two dimensional array)** .

MARK [2, 4]

المادة / اسم التلميذ	اللغة العربية	الدين	اللغة الإنجليزية	اللغة الفرنسية	الرياضة	الطبيعة
محمد سمير	80	90	82	75	98	79
رانى محمد	80	95	90	76	100	81
سالى أسامة	75	80	90	83	99	85
مها حسن	79	85	85	82	85	80
عمرو الحسينى	85	74	92	80	100	65

شكل (٧ ــ ٢)

(١) **كما يطلق عليها الرياضيون الاسم** (Matrix) بمعنى مصفوفة أيضاً . وإذا كان عدد الأعمده مساوياً لعدد الصفوف فإنها تسمى المصفوفة المربعة (square matrix) .

وقد مررنا بمثل هذه الحالة فى الأبواب السابقة ولكننا عالجناها بطرق بدائية كتدريب على استخدام الحلقات التكرارية . فإذا تعاملنا معها كمصفوفة فإن لغة باسكال تمدنا بتسهيلات جديدة لبرمجة المسألة .

ويمكننا عندئذ أن نمثل درجة أى تلميذ عام فى أى مادة عامة بالمتغير :

MARK[I,J]

وقد استخدمنا هنا دليلين ا , ل حيث يدل ا على اسم التلميذ (الصف) وتدل ل على المادة (العمود) ، وبذلك تكون MARK[1,1] هى درجة التلميذ الأول فى المادة الأولى أما mark[1,5] فهى درجة التلميذ الأول فى المادة الخامسة ، كذلك فإن MARK[4,2] هى درجة التلميذ الرابع فى المادة الثانية .. وهكذا ..

وكما نرى فإنه فى حالة تمثيل المصفوفة ذات البعدين فإن المتغير الدليلى يحتاج إلى دليلين حتى يعبر عن أى عنصر من عناصر المصفوفة .

والدليل فى لغة باسكال لا يشترط أن يكون حرفاً واحداً مثل ل,ا ولكن ينطبق عليه ما ينطبق على أسماء البيانات من قواعد . فالأدلة الآتية جائزة أيضاً :

mark[student, subject]

والمصفوفة ذات البعدين هى أيضاً فئة لأنها تحتوى على عناصر تنتمى انتماءً واحداً . وكلمة الفئة هى كلمة مألوفة فى الرياضة الحديثة .

(٧ ــ ٢) *الإعلان عن المصفوفات فى البرنامج :*

يتم الإعلان عن المصفوفات بطريقة خاصة تؤخذ فيها عدة اعتبارات . فقد تحتوى المصفوفة على أرقام حقيقية أو صحيحة وقد تحتوى على لبنات وهذا يستلزم الإعلان عنه فى بداية البرنامج .

كما أن الإعلان عن المصفوفة يعنى حجز عدد من الأماكن فى الذاكرة يتسع لعناصر المصفوفة أى يساوى بُعد المصفوفة . لذلك يلزم التنويه عن نوعية المتغيرات الدليلية وعن عددها .

ولنضرب مثلاً بالمصفوفة الآتية التى تعبر عن أسعار ثمانية أصناف مختلفة من السلع شكل (٧ — ٣) ، وسوف نفترض أن اسم المتغير الدليلى العام هو :

priceof[i]

اسم المتغير	المحتوى (السعر)
priceof[1]	5.77
priceof[2]	3.15
priceof[3]	2.50
priceof[4]	16.33
priceof[5]	2.50
priceof[6]	13.45
priceof[7]	.86
priceof[8]	1.98

شكل (٧ — ٣)

للإعلان عن مثل هذه المصفوفة فى بداية البرنامج نكتب السطر التالى :

```
var  priceof : array [1..8] of real;
```

لاحظ الكلمات الجديدة التى تحتها خط فى هذا الإعلان !

إن هذا الإعلان يحتوى على كلمة array بمعنى مصفوفة ويحتوى على أبعاد المصفوفة أو عدد عناصرها وهو ثمانية عناصر ، كما يحتوى على نوع المتغيرات وهو النوع الحقيقى (real) .

والطريقة التى تستخدم للتعبير عن بُعد المصفوفة طريقة مميزة فهى تحتوى على رقم أول عنصر (1) وآخر عنصر (8) والنقطتان بينهما تعنيان بقية عناصر الفئة . وتتميز هذه الطريقة بإمكان إنشاء مصفوفة تبدأ من أى عنصر آخر خلال العنصر رقم 1 ، فيجوز أن نبدأ بالعنصر الخامس مثلاً فيصبح بعد المصفوفة **[5..8]** وهى فى هذه الحالة تحتوى على ٤ عناصر فقط .

أما الإعلان عن المصفوفة ذات البعدين فهو يتم بطريقة مماثلة فالمصفوفة الموضحة شكل (٧ ــ ٤) تحتوى على ٨ صفوف و ١٠ أعمدة أى أن أبعادها ٨×١٠ بمعنى أنها تتسع لثمانين عنصر . فإذا أطلقنا على المتغيّر العام لهذه المصفوفة الاسم .

popmap[row,column]

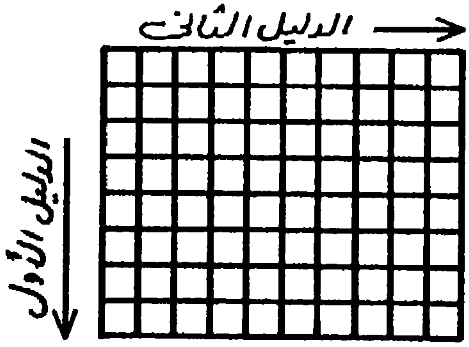
حيث يمثل الدليل row رقم الصف .

ويمثل الدليل column رقم العمود .

فإن الإعلان يكون بالطريقة الآتية :

var popmap : array [1..8,1..10] of integer;

وهى مصفوفة من الأعداد الصحيحة .



شكل (٧ ــ ٤)

(٧ ــ ٣) معالجة المصفوفات ذات البعد الواحد

One-dimensionl arrays

بعد الإعلان عن المصفوفة يصبح لدينا عدد من المتغيّرات الدليلية مساوٍ لعدد عناصر المصفوفة ولنا أن نجرى على هذه المتغيرات ما نشاء من عمليات المعالجة التى نجريها على المتغيرات العادية من التخصيص والقراءة والطباعة إلى آخره ، ولكن علينا ألا ننسى أثناء كتابة البرنامج أن نكتب الدليل المناسب للمتغير كالمثال الآتى :

مثال (٧ ــ ١) :

```
read(price[1], price[2], price[3], price[4]);
read(price[5], price[6], price[7], price[8]);         ← قراءة
writeln('the price of item 2 is ',price[2]);
writeln('the price of item 4 is ',price[4]);          ← طباعة

writeln('the price difference between items 1 and 2 is ',
         abs(price[1] - price[2]));

difference := price[1] - price[2];          ← معالجة حسابية
if difference > 0 then
   writeln('item 1 costs ', difference,
          ' more than item 2.')
else   if difference < 0 then
        writeln('item 2 costs ', -difference,
                ' more than item 1.')
```

شكل (٧ ــ ٥)

وللحلقات التكرارية فائدة كبيرة فى معالجة المصفوفات لا سيما فى القراءة والطباعة . فبدلاً من قراءة كل عنصر باستخدام اسمه الخاص يمكن استخدام الاسم العام بداخل حلقة تكرارية بحيث يكون الدليل هو نفسه عدّاد الحلقة التكرارية كالمثال الآتى : شكل (٧ ــ ٦) :

مثال (٧ ـــ ٢) :

```
program reverse(input,output);

var number : array [1..10] of integer;
    position : integer;

begin

   for position := 1 to 10 do
      read(number[position]);

   for position := 10 downto 1 do
      writeln(number[position])

end.
```

شكل (٧ ـــ ٦)

والبرنامج السابق يقوم بقراءة مصفوفة ذات بعد واحد مكوّنة من عشرة عناصر من الأعداد الصحيحة ، ثم يطبع المصفوفة بطريقة عكسية أى بدءاً من العنصر العاشر وحتى العنصر الأول .

مثال (٧ ـــ ٣) :

فى بعض الأحيان تصبح عملية التخصيص ذات فائدة لا سيما عندما تكون محتويات المتغيرات ثابتة . والبرنامج التالى شكل (٧ ـــ ٧) يوضح كيفية حساب سعر كميّة معينة من الأصناف المباعة . فهو يحتوى على أسعار أربع سلع مختلفة تم تخصيصها للمتغيرات الدليلية الممثلة لأسعار الأصناف ... **priceof[1], priceof[2]** ويستقبل البرنامج الكمية المباعة quantity ثم يحسب سعرها ويطبعه .

```
program prices(input,output);

var priceof : array [1..4] of real;
     itemno, quantity : integer;
     total : real;

begin

   priceof[1] := 5.77;
   priceof[2] := 3.15;
   priceof[3] := 2.50;
   priceof[4] := 1.35;

   total := 0;

   for itemno := 1 to 4 do
   begin
      read(quantity);
      total := total + quantity*priceof[itemno]
   end;

   writeln('the total cost is ',total)
end.
```

التخصيص لأسعار الأصناف

حساب السعر الكلي

شكل (٧ — ٧)

وبالطبع فإن طريقة التخصيص ليست هى الطريقة المثلى حيث أن أى تغيير يطرأ على سعر السلعة سوف يستلزم إصلاح البرنامج نفسه . أما المتبع فى مثل هذه الحالات أن تحفظ أسعار السلع فى ملف (file) وتتم القراءة منه آلياً . وفى حالة تغيير الأسعار فإن برنامجاً آخر يتولى **تحديث الملف** .

مثال (٧ — ٤) :

رأينا من قبل المصفوفة ذات الأعداد الحقيقية أو الأعداد الصحيحة ، وهذا المثال يعرض نموذجاً لمصفوفة اللبنات . شكل (٧ — ٨) .

ونلاحظ فى هذا البرنامج أن عدد عناصر المصفوفة المعلن عنها هو ٢٠

عنصراً . لكن البرنامج يقرأ ويطبع عدداً من العناصر يتحدد بالمتغيّر n أثناء تنفيذ البرنامج .

فالبرنامج يبدأ بقراءة قيمة المتغير n بموجب العبارة **readln(n)** فإذا أدخلنا له العدد 5 مثلاً فسوف يقوم البرنامج بمعالجة خمسة عناصر فقط من المصفوفة . ويجوز إدخال أى رقم كقيمة للمتغير n بشرط ألاّ يزيد عن **20** وإلاّ يفشل البرنامج .

```
program word(input,output);

var letter : array [1..20] of char;
    n, i : integer;

begin

  readln(n);
  for i := 1 to n do
     read(letter[i]);

  writeln;
  for i := n downto 1 do
     write(letter[i])

end.
```

شكل (٧ — ٨)

مثال (٧ — ٥) تجميع وفرز الأصوات :

نعود مرة أخرى إلى معالجة الانتخابات ونلتقى فى هذا المثال بستّة من المرشحين ويتولى البرنامج فرز الأصوات التى حصل عليها المرشحون وإعلان رقم المرشح الفائز . شكل (٧ — ٩) .

تجميع وفرز الأصوات

```
program votes(input,output);

var candidate, mostvotesofar, winner : integer;
    votesfor : array [1..6] of integer;

begin
```

شحن المتغيرات بقيمة ابتدائية صفراً

```
   for candidate := 1 to 6 do
      votesfor[candidate] := 0;
```

إدخال البيانات

```
   read(candidate);
   repeat
      votesfor[candidate] := votesfor[candidate] + 1;
      read(candidate);
   until candidate = -1;
```

نهاية البيانات

فرز الأصوات

```
   mostvotesofar := 0;
   for candidate := 1 to 6 do
      if votesfor[candidate] > mostvotesofar then
         begin
            mostvotesofar := votesfor[candidate];
            winner := candidate
         end;

   writeln('winner is candidate no. ',winner);
end.
```

رقم المرشح الفائز

شكل (٧ ــ ٩)

وقد استخدمت فى هذا البرنامج الأرقام من 1 إلى 6 كأرقام مرجعية للدلالة على كل مرشح. وقد استخدم المتغيّـر العـام **votesfor[candidate]** للتعبير عن المتغيرات الدليلية لمصفوفة المرشحين فمثلاً:

هو متغيّر تجميع الأصوات للمرشح الأول **votesfor[1]**

هو متغيّر تجميع الأصوات للمرشح الثانى **votesfor[2]**

وهكذا ...

يبدأ البرنامج بالإعلان عن المتغيّرات المستخدمة وهى :

(١) متغيرات صحيحة (integer) :

- candidate رقم المرشح (وهو يستخدم كدليل لمتغيرات المصفوفة) .
- mostvotesofar وعاء لتجميع أعلى نسبة أصوات لأى مرشح .
- winner رقم المرشح الفائز .

(٢) المصفوفة (array) :

تم الإعلان عن مصفوفة واحدة ذات عناصر ستة وهى : votesfor وهى تستخدم المتغير candidate للدلالة على رقم المرشح .

وينقسم البرنامج إلى ثلاثة أجزاء أساسية تم وضعها بداخل مستطيلات للإيضاح :

- **الجزء الأول (المستطيل الأول) :**

لشحن جميع المتغيرات الدليلية بالقيمة الابتدائية « صفر » وهذه عادة حميدة يجب إتباعها مع كل البرامج .

- **الجزء الثانى (المستطيل الثانى) :**

لإدخال الأصوات التى حصل عليها المرشحون ، والنظام المتبع هنا هو إدخال رقم المرشح الذى حصل على صوت ، فالرقم 1 يدل على أن المرشح الأول حصل على صوت واحد ، والرقم 5 يدل على أن المرشح الخامس حصل على صوت واحد وهكذا ..

وبمجرد إدخال صوت واحد فإنه يضاف إلى الأصوات السابقة التى حصل عليها نفس المرشح .

فإذا انتهت جميع الأصوات ندخل إلى البرنامج العدد (1 –) للدلالة على انتهاء

إدخال البيانات .

أما الجزء الثالث (المستطيل) : فهو يستخدم لمقارنة محتويات العناصر الستة للمصفوفة لإيجاد أكبر نسبة من الأصوات حصل عليها أى مرشح .

والمنطق المستخدم للمقارنة منطق معروف ، وهو مقارنة كل متغير مع الوعاء mostvotesofar فإذا كان محتوى المتغيّر أكبر من محتوى الوعاء يتم وضع محتوى المتغيّر فى الوعاء ، وبذلك فإن الوعاء يحتوى دائماً على أكبر رقم . أما دليل المتغير الذى تجرى عليه المقارنة فهو يوضع فى الوعاء winner . وفى نهاية عملية المقارنة سوف يحتوى الوعاء winner على رقم المرشح الذى حصل على أعلى نسبة أصوات .

مثال (٧ ــ ٦) عدد أيام الإجازة :

عالجنا من قبل موضوع « كم تكلفك الإجازة ؟ » فى الباب السادس . وفى هذا المثال نضيف إلى هذا المنطق معالجة جديدة باستخدام المصفوفات لحساب الفترة بالأيام بين تاريخين متتاليين يتكون كل منهما من اليوم والشهر .

```
program days(input,output);

var month, daystogo, firstday,
    firstmonth, secondday, secondmonth : integer;
    daysin : array [1..12] of integer;

begin
   for month := 1 to 12 do                    عدد الأيام في كل شهر

      case month of
                     2: daysin[month] := 28;
              9,4,6,11: daysin[month] := 30;
      1,3,5,7,8,10,12: daysin[month] := 31
      end;
                                   التاريخان في نفس الشهر
   read(firstday,firstmonth,secondday,secondmonth);
   if firstmonth = secondmonth then
      daystogo := secondday - firstday
   else
   begin                           التاريخان في شهرين مختلفين
      daystogo := daysin[firstmonth] - firstday;

      for month := firstmonth + 1 to secondmonth - 1 do
      daystogo := daystogo + daysin[month];

      daystogo := daystogo + secondday
   end;                                           عدد الأيام

   writeln('days to go:', daystogo)

end.
```

شكل (٧ — ١٠)

متغيرات البرنامج :

(١) المتغيرات الصحيحة :

- month : أحد شهور السنة وهو يأخذ القيم من 1 إلى 12 ويستخدم كدليل للمصفوفة daysin .
- daystogo : عدد الأيام فى الفترة المطلوبة بين التاريخين (أيام الإجازة)
- firstday : رقم اليوم الأول } بداية المدة
- firstmonth : رقم الشهر الأول }
- secondday : رقم اليوم الثانى } نهاية المدة
- secondmonth : رقم الشهر الثانى }

(٢) المصفوفات :

- daysin مصفوفة صحيحة تحتوى على ١٢ عنصراً

والمتغيّر الدليلى العام لهذه المصفوفة هو **daysin[month]** ويعنى عدد الأيام فى شهر ما مثل :

daysin[2]: = 28 عدد أيام شهر فبراير
daysin[3]: = 31 عدد أيام شهر مارس

والبرنامج يقوم بحساب عدد الأيام بين التاريخين وفقاً لحالتين :

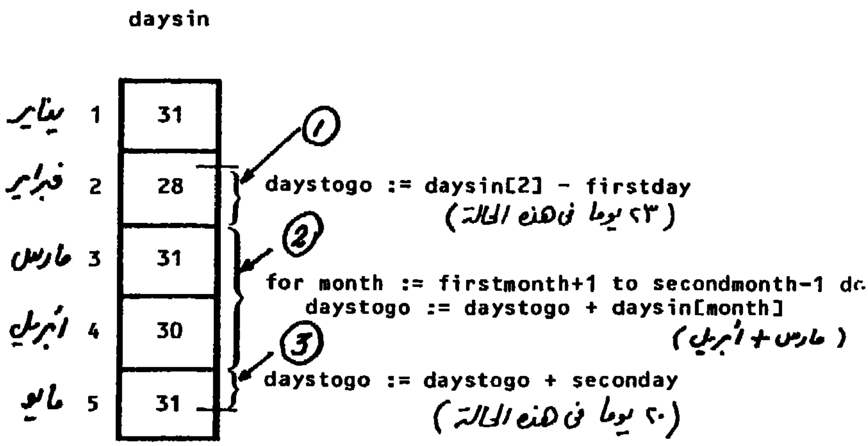
(١) إذا كان التاريخان فى نفس الشهر فيتم ذلك بطرح رقم اليوم الأول من رقم اليوم الثانى .

(٢) إذا كان الشهران مختلفين فإن يتم جمع ثلاثة أجزاء معاً لتكوين عدد الأيام المطلوبة . فإذا كان التاريخان على سبيل المثال هما :

5 2 (٥ فبراير)

20 5 (٢٠ مايو)

فإن المدة بينهما تحسب بإضافة الأجزاء الثلاثة الموضحة فى شكل (٧ — ١١) .



شكل (٧ — ١١)

(٧ — ٤) الرسم البيانى ***histograms*** :

عالجنا من قبل موضوع رسم الأشكال المختلفة باستخدام الحلقات التكرارية والرسم البيانى أو الهستوجرام histogram يعنى تحويل المعلومات الناتجة من الكومبيوتر من صورتها العددية إلى رسم بيانى يسهل فهمه بإلقاء نظرة عامة عليه بدلاً من فحص الأرقام . والرسم البيانى ذو أهمية كبيرة فى الموضوعات الإحصائية .

```
          total sales for each dept.(div. by 10)
          --------------------------------------
                            المبيعات الكلية للأقسام المختلفة
            1 *************
            2 ***********************************
            3 ************************
dept. no.   4 ********************************
رقم القسم    5 *********
            6 ********
            7 ****************************************
            8 *****
```

شكل (٧ — ١٢)

فإذا ألقينا نظرة عامة إلى شكل (٧ — ١٢) الذى يوضح المبيعات الكلية لثمانية أقسام فى شركة ما ، فسوف ندرك على الفور أن القسم رقم 7 هو الذى حقق أعلى نسبة مبيعات وأن القسم رقم 8 هو أقل الأقسام فى المبيعات .

ولعل المشاهد لهذا الرسم يستطيع الخروج بانطباع سريع عن نشاط كل قسم من الأقسام بالنسبة لبقية الأقسام الأخرى ، وهذا لا يتحقق عادة إذا كان الخرج عبارة عن أرقام تحتاج للفحص والمقارنة .

والبرنامج التالى يوضح كيفية الحصول على هذا الرسم البيانى باستخدام برنامج باسكال .

مثال (٧ — ٧) تحليل المبيعات بالرسم البيانى :

```
program salesanalysis2(input,output);

var dept, col, deptsales, salesin10s : integer;

begin
   writeln('total sales for each dept.(div. by 10)' : 48);
   writeln('--------------------------------------' : 48);
   for dept := 1 to 8 do
   begin
      if dept = 4 then write('dept. no. 4 ')
      else           ,  write(dept:11, ' ');

      read(deptsales);
      salesin10s := deptsales div 10;
      for col := 1 to salesin10s do
         write('*');
      writeln
   end

end.
```

البرنامج

طباعة أرقام الأقسام

مقياس الرسم

الرسم

تحليل المبيعات بالرسم

شكل (٧ ــ ١٣)

والمتغيرات المستخدمة فى هذا البرنامج هى :

- dept : رقم القسم (العدّاد) .
- col : رقم العمود الرأسى فى خانات الطباعة (العدّاد) .
- deptsales : مبيعات القسم
- **salesin10s** : مبيعات القسم مقسومة على مقياس الرسم **(10)** .

وكلها متغيرات صحيحة .

ومقياس الرسم ضرورى فى مثل هذه البرامج حتى لا يخرج الرسم عند حدود الورقة ويترك تقديره للمبرمج بحسب اتساع خط الطباعة .

هل لاحظنا أن البرنامج لم يستخدم المصفوفات ؟

فى الواقع أنه لم تكن هناك حاجة لاستخدامها فالبيان تتم قراءته ومعالجته فور إدخاله .

ولكن فى بعض الأحيان قد يصبح استخدام المصفوفات ضرورياً ولنفرض هذه الحالة عندما يكون اتساع خط الطبع ٦٠ لبنة وعندما لا يمكن التنبؤ إذا ما كانت أعلى نسبة للمبيعات سوف تتعدى هذا الاتساع أم لا . فى هذه الحالة يلزم إجراء معالجة بهدف معرفة مقياس الرسم المناسب . فيصبح البرنامج كالآتى :

```
program salesanlaysis3(input,output);

const lengthofline = 60;
var salesfor : array [1..8] of integer;
 dept, col,  maxsofar, scaling, scaledtotal : integer;

begin
   maxsofar := 0;
   for dept := 1 to 8 do
   begin
      read(salesfor[dept]);
      if salesfor[dept] > maxsofar then
            maxsofar := salesfor[dept]
   end;

   scaling := 1 + maxsofar div lengthofline;

   for dept := 1 to 8 do
   begin
      write(dept, ' ');
      scaledtotal := salesfor[dept] div scaling;
      for col := 1 to scaledtotal do
         write('*');
      writeln
   end

end.
```

مصفوفة الأقسام

تحديد أعلى نسبة مبيعات

تحديد مقياس الرسم

الرسم

تحليل المبيعات بالرسم

شكل (٧ — ١٤)

(٧ ــ ٥) شحن المتغيرات بقيمة ابتدائية **initialization** :

لعلنا لاحظنا فى البرامج السابقة جميعاً أنه يتم شحن المتغيرات دائماً بقيمة ابتدائية صفراً (ما لم تكن هناك قيمة ابتدائية خلاف الصفر) ، وهذه الخطوة تعتبر ضرورية فى البرنامج لأن خانات الذاكرة فى بعض الأحيان قد تحتوى على « مخلّفات » من البيانات ولنر هذا المثال الذى نقرأ فيه مصفوفة عددية ونشحنها بالقيمة صفراً ثم نطبعها شكل (٧ ــ ١٥) .

البرنامج

```
program chararrays(input,output):
VAR ch: array[1..20] of integer;
    i:integer;

BEGIN

    for i:=1 to 10 do      ← الشحن الابتدائى
    ch[i]:=0;

    writeln;
    for i:=1 to 10 do      ← الطباعة
    writeln(ch[i]);

END.
```

التنفيذ

```
Running
0
0
0
0
0
0
0
0
0
0
```

عناصر المصفوفة

شكل (٧ ــ ١٥)

لنحذف الآن عبارة شحن المتغيّرات بقيمة ابتدائية بوضعها بين القوسين "{ }" اللذين يحوّلانها إلى ملاحظة ، ولنر النتيجة .

```
program chararrays(input,output);
VAR ch: array[1..20] of integer;
    i:integer;

BEGIN
    { for i:=1 to 10 do
    ch[i]:=0.0;}
    writeln;
    for i:=1 to 10 do
    writeln(ch[i]);

END.

Running

0
0
0
0
0
0
0
0
-32768
358
```

(ملاحظات بخط اليد: عند السطرين المحاطين: «عبارة محذوفة (لاحظ أقواس التعليقات)»؛ عند القيمتين الأخيرتين: «مخلفات بيانات!»)

شكل (٧ ــ ١٦)

ها هى النتيجة تحتوى على مخلفات لا معنى لها من الأرقام .

ولو قمنا بطباعة عشرين عنصراً من المصفوفة ربما نصادف المزيد من

الفضلات كما فى شكل (٧ ــ ١٧) .

```
.
Running

0
0
0
0
0
0
0
0
-32768
358
5632
12336
12336
12336
12336
12336
48
358
5682
0

>
```

مخلفات بيانات

شكل (٧ ــ ١٧)

لذلك يجب « تصفير » جميع المتغيرات قبل التعامل معها حتى لا تتسبب فى الحصول على نتائج خاطئة .

(٧ ــ ٦) معالجة المصفوفات ذات البعدين :

يمثل شكل (٧ ــ ١٨) مصفوفة ذات بعدين مكونة من ثمانية صفوف وعشرة أعمدة ، فإننا عادة نقرأ المصفوفة صفاً صفاً من خلال حلقة تكرارية كالآتى :

for row := 1 to 8do

...

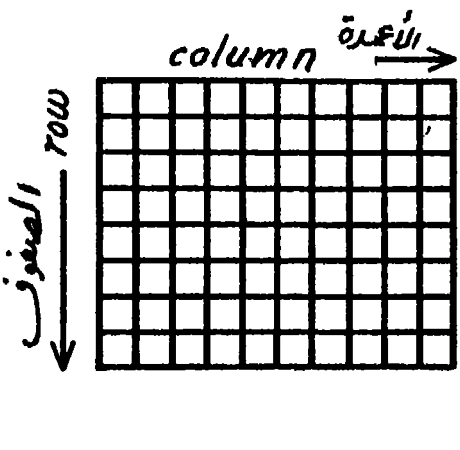
(عملية قراءة أو معالجة الصف) ...

ولكن عملية قراءة أو معالجة صف واحد تتضمن معالجة الأعمدة العشرة التى تساهم فى تكوين كل صف . لذلك فإن عملية القراءة للصف الواحد سوف تكون عبارة عن حلقة تكرارية أخرى بداخل الحلقة الأولى كالآتى :

for column := 1 to 10 do

...

(عملية قراءة أو معالجة لكل عنصر)



شكل (٧ ــ ١٨)

ولنر هذا المثال :

مثال (٧ ـــ ٨) تعدّاد السكان :

لو اعبترنا أن كل مربع صغير فى شكل (٧ ـــ ١٨) من المربعات الثمانين يمثل منطقة من المناطق السكنية على خريطة المدينة ، فيمكن اعتبار أن الشكل كله يصوّر لنا تعدّاد السكان فى مدينة ما .

فإذا أردنا الإعلان عن هذه المصفوفة فلنمنحها اسماً مثل popmap بمعنى خريطة التعداد .

وللإعلان عن المصفوفة نكتب السطر التالى :

```
var popmap : array [1..8,1..10] of integer;
```

ويصبح بذلك المتغيّر الدليلى العام هو :

```
popmap[row, column]
```

حيث row هو رقم الصف .

column هو رقم العمود .

والبرنامج التالى يستخدم هذه المصفوفة فى حساب التعداد الكلى للسكان (total) ويطبع هذا التعداد علاوة على تعداد كل منطقة من المناطق . شكل (٧ ـــ ١٩) ويبدأ البرنامج بقراءة المصفوفة .. وهذا يستلزم منا كتابة ثمانين رقماً مختلفاً إذا كنا سوف نستخدم لوحة الأزرار .. ونلاحظ فى تنفيذ البرنامج أن الأرقام المدخلة لا تحتوى أية فواصل فيما بينها ولكن كل رقم مع ذلك يعقبه الضغط على زر الإدخال (ENTER) . وقد استخدمنا أرقاماً صغيرة بين الصفر والتسعة لتمثيل عدد السكان بالآلاف فى كل منطقة لتسهيل عملية إدخال البيانات .

أما الجزء الثانى من البرنامج فهو تجميع العناصر كلها فى المتغير : total وطباعته .

والجزء الأخير هو طباعة عناصر المصفوفة كلها تمثيل عدد السكان فى كل منطقة .

★ ★ ★

تعداد السكان

قراءة المصفوفة ← BEGIN

جمبع عناصر المصفوفة ← total:=0;

طباعة التعداد الكلى ← writeln('total population is ',total);

طباعة عناصر المصفوفة ← for row :=1 to 8 do

```
program population(input,output);
VAR popmap: array[1..8,1..10] of integer;
    row,column,total:integer;

BEGIN
    for row :=1 to 8 do
      for column :=1 to 10 do
        read(popmap[row,column]);

    total:=0;
    for row :=1 to 8 do
      for column :=1 to 10 do
        total:=total+popmap[row,column];

    writeln;
    writeln('total population is ',total);
    writeln('populations of individual zones are:');

    for row :=1 to 8 do
      BEGIN
       writeln;
         for column :=1 to 10 do
           write(popmap[row,column]);
    END;
```

تنفيذ البرنامج :

```
Running
98999766578694319122223346567326788965433222221118998586543233355454545667656 76
total population is 399
populations of individual zones are:

9899976657
8694319122
2233465673
2267889654
3322222111
8998586543
2333554545
4566765676
>
```

إدخال ٨٠ عنصراً

التعداد الكلى

عناصر المصفوفة

مثال (٧ ـــ ٩) .. تحليل تعداد السكان :

فى هذا البرنامج يتم قراءة عناصر المصفوفة الممثلة لتعداد السكان كما فى المثال السابق ثم يقوم البرنامج بإيجاد المناطق التى يقل تعداد السكان فيها عن نصف متوسط تعداد السكان فى المناطق المحيطة بها من الجهات الأربع (الشمال والجنوب والشرق والغرب) .

وحساب المتوسط يتم بإضافة تعداد السكان فى الأربعة مربعات المحيطة بكل مربع وقسمة الناتج على أربعة . فإذا كانت إحداثيات أى مربع عام هى : row, col فإن الإحداثيات الآتية تمثل المربعات الأربعة المحيطة :

- المربع الشرقى **row, col+1**
- المربع الغربى **row, col−1**
- المربع الشمالى **row−1, col**
- المربع الجنوبى **row+1, col**

```
program popmap3(input,output);

var popmap : array [1..8,1..10] of integer;
    row, col : integer;  average : real;

begin
   for row := 1 to 8 do
      for col := 1 to 10 do
      read(popmap[row,col]);

   for row := 2 to 7 do
      for col := 2 to 9 do
      begin
         average := (popmap[row-1,col]
                    +popmap[row+1,col]
                    +popmap[row,col-1]
                    +popmap[row,col+1])/4;
         if popmap[row,col] < average/2
         then writeln('the region ', row, col,
                      ' is underpopulated')
      end
end.
```

شكل (٧ ـــ ٢٠)

(٧ ــ ٧) ملاحظات على المتغيّرات الدليلية :

علمنا من قبل أن دليل المتغيّر فى مصفوفة ما يأخذ قيماً تبدأ من الواحد وتنتهى بالعدد الذى يعبر عن بُعد المصفوفة . كما علمنا أنه من الجائز أن يبدأ الدليل من قيمة أخرى خلاف الواحد . وبصفة عامة يمكن أن يكون الدليل **ثابتاً أو ثابتاً مسمى .**

ولنر هذا المثال حيث يعرض علينا حالة إحدى الشركات التى تستخدم مسلسلاً رقميّاً لتشفير السلع التى تبيعها ، وتقع الأرقام المسلسلة فى النطاق من **3001** إلى **3563** . والبرنامج يقرأ قائمة الأسعار لهذه الأصناف ، ثم يقرأ الكمية المطلوبة من كل صنف ويحسب السعر الكلى للطلبية . وتستخدم علامة النجمة * لإنهاء إدخال البيانات .

مثال (٧ ــ ١٠) حساب المبيعات :

```
program sale(input,output);

const minref = 3001; maxref = 3563;
var priceof : array [minref..maxref] of real;
    ref, nextref, quantity : integer;
    totalprice : real; nextch : char;

begin
   for nextref := minref to maxref do
      read(priceof[nextref]);

   totalprice := 0;
   repeat
      read(ref,quantity);
      totalprice := totalprice + priceof[ref]*quantity
      read(nextch)
   until nextch = '*';

   writeln('total cost of order is ', totalprice)
end.
```

شكل (٧ ــ ٢١)

مثال (٧ ــ ١١) معدّل تكرار الحروف فى نص :

هل يمكن استخدام اللبنات كأدلّة للمصفوفة ؟

فى هذا المثال نعرض تطبيقاً لاستخدام الحروف الأبجدية (اللبنات) كأدلة لمصفوفة من الأعداد الصحيحة . ويقوم البرنامج بحساب معدّل تكرار كل حرف فى قطعة من نص تنتهى بالعلامة (*) ويتم التعبير عن معدّل التكرار بتوقيعه كرسم بيانى .

ويبدأ البرنامج بشحن المتغيرات الدليلية بالقيمة « صفر » ثم يقرأ لبنة ويختبر ما إذا كانت تنتمى إلى الفئة ['a'..'z'] أى فئة الحروف الأبجدية ، وفى هذه الحالة يضيف واحداً إلى عنصر المصفوفة الذى يمثل عدّاد هذه اللبنة . فإذا كان المتغيّر الدليلى **lettercount['a']** يحتوى على العدد 5 كان معنى ذلك أن الحرف **'a'** تكرر خمس مرات وهكذا بالنسبة لبقية الحروف .

```
program freqcount(input,output);                      اللبنات كأدلة

var lettercount : array ['a'..'z'] of integer;
    letter, character : char; col : integer;
                                                      اللبنات كعداد
begin
   for letter := 'a' to 'z' do
      lettercount[letter] := 0;

   read(character);                                   حساب معدل تكرار حرف
   repeat
      if character in ['a'..'z'] then
   lettercount[character]:=lettercount[character]+1;
read(character);
   until character = '*';

   for letter := 'a' to 'z' do
   begin
      writeln;                                        رسم الهستوجرام
      for col := 1 to lettercount[letter] do
         write('*')
   end

end.
```

شكل (٧ ــ ٢٢)

والمتغيرات المستخدمة هى :

- lettercount مصفوفة أعداد صحيحة ذات أدلة من الحروف الأبجدية تستخدم لحساب معدّل تكرار كل حرف .
- letter متغير لبنة يحتوى على أحد الحروف الأبجدية ويستخدم كدليل للمصفوفة .
- character متغير لبنة يحتوى على أحد لبنات النص تحت المعالجة .
- col متغير صحيح يمثل العمود الرأسى (الخانة) المستخدم فى توقيع الرسم .

وهذا البرنامج لا يعدّ ذات قيمة عملية إلاّ إذا تمت قراءة النص من ملف .

٧ ـ ٨) الأنماط المبتكرة *Type* :

عندما تحدثنا عن الأنماط من قبل عرضنا أنواعاً أربعة من الأنماط وهى : الصحيحة والحقيقية والمنطقية وأنماط اللبنات .

Integer
Real
Boolean
Char

ومع ذلك ففى بعض الأحيان قد نحتاج لتمثيل أنواع من البيانات لا تندرج تماماً تحت هذه الأنماط . وعلى سبيل المثال إذا أردنا تمثيل عدد الدقائق الزمنية الموجودة فى كسر الساعة (وهى تقع فى النطاق من 1 إلى 59) فمن المنتظر فى حدود ما عرفناه حتى الآن أن نعلن عن متغير مثل minute من النوع الصحيح integer .

وعلى المبرمج فى هذه الحالة أن يمد برنامجه بالوسائل اللازمة لحمايته من البيانات غير الصحيحة مثل العدد 82 أو العدد السالب 28 .

هناك وسيلة أكثر سهولة تحققها لنا لغة باسكال باستخدام **فئة « جزئية » من الأعداد الصحيحة فقط .**

(٧ ـ ٨ ـ ١) أنماط الفئات الجزئية (subranges) :

يمكننا أن نُعرّف الفئة الجزئية للأعداد الصحيحة التى تمثل الدقائق (كسور الساعة) كالآتى :

VAR minutes : 1..59;

معنى ذلك أن المتغير minutes لا يجوز أن يحتوى على أى قيمة صحيحة خارج نطاق الأعداد من 1 إلى 59 ولذلك يطلق عليه الاسم **الفئة الجزئية** (subrange) ، حيث أنه يعتبر فئة جزئية من فئة الأعداد الصحيحة كلها .

ويجوز الإعلان عن أية فئات جزئية مادامت فئة ذات **عدد محدود من العناصر** . (الأرقام الحقيقية لا تخضع لهذا التعريف لأن الأرقام التى تنحصر بين الرقمين 5 ، 7 مثلاً لا يمكن حصرها) لكنه من الجائز إنشاء فئة جزئية من الحرفيّات كالمثال الآتى :

VAR numerics :'0'..'9';

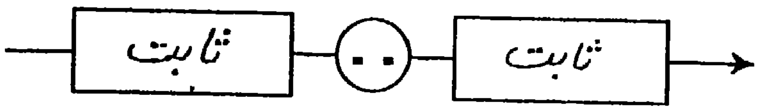
ولاستخدام الفئات الجزئية عدة مميزات منها :

(١) منع البيانات غير الصحيحة من المعالجة بواسطة البرنامج حيث يقوم البرنامج بإرسال رسالة الخطأ المناسبة .

(٢) تقليل حجم الذاكرة المخصصة لتمثيل البيانات بالأرقام الثنائية .

والشكل (٧ ـ ٢٣) يوضح قاعدة تكوين نمط الفئات الجزئية

بالرسم .



شكل (٧ ــ ٢٣)

(٧ــ٨ــ٢) أنماط القوائم **Enumeration types** :

فى الكثير من لغات الكومبيوتر قد نحتاج إلى تمثيل أيام الأسبوع بالأرقام وذلك بإعطاء كود لكل يوم مثل :

monday = 0 .... sunday = 6

وهذا يقابله فى لغة باسكال إعلان المتغير dayofweek كفئة جزئية من 0 إلى 6 كالآتى :

VAR dayofweek : 0..6

وبذلك يمكن أن يتضمن البرنامج عبارة مثل :

```
IF (dayofweek = 5) OR (dayofweek = 6)
   THEN
```

بمعنى إذا كان اليوم هو يوم السبت (رقم 5) أو يوم الأحد (رقم 6).....

كما يجوز إعلان أيام الأسبوع كثوابت مسماه كالآتى :

```
CONST mon  = 0;  tues = 1;  wed  = 2;  thurs =  3;
  f r i  =  4;  sat  =  5;  sun  =  6;
```

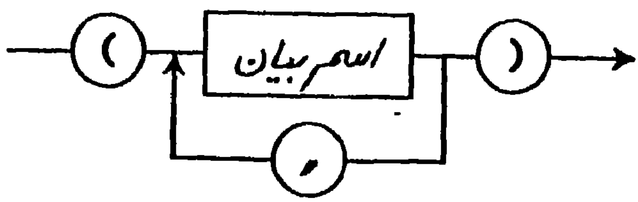
وفى هذه الحالة فإننا يمكن أن نستخدم الحروف بدلاً من الأرقام فى التعامل مع الأيام خلال البرنامج إذ يمكننا كتابة العبارة الشرطية السابقة بصورة أفضل وأكثر وضوحاً للقارىء :

```
IF (dayofweek  =  sat) OR (dayofweek  =  sun)
                                      THEN ...'
```

ومع ذلك فكل هذه المحاولات تحتاج جهداً فى البرمجة ، ويمكننا الاستغناء عنها جميعاً باستخدام **نمط القائمة** كالآتى :

```
VAR dayofweek : (mon,tues,wed,thurs,fri,sat,sun);
```

وقاعدة تكوين نمط القائمة موضحاً بالرسم فى شكل (٧ــ٢٤) .



شكل (٧ ــ ٢٤)

هذا التعريف يتضمن تلقائياً تمثيل الأيام بأرقام كودية من 0 إلى 6 كخاصية من خصائص اللغة دون أن يشعر المبرمج ودون أن يبذل جهداً فى ذلك . وفى هذه الحالة يمكننا استخدام نفس العبارة الشرطية السابقة باستخدام الثوابت

الواردة فى الإعلان :

IF (dayofweek = sat) OR (dayofweek = sun)
THEN...

ومن البديهى أنه لا يجوز استخدام الثوابت العددية للتخصيص للمتغير dayofweek فى هذه الحالة ؛ فالعبارة الآتية تصبح عبارة خاطئة :

IF (dayofweek = 5) OR (dayof week = 6)
THEN

وذلك لأن الأعداد 5 ، 6 ليست متضمنة فى القائمة المعلن عنها صراحة . كذلك لا يجوز إجراء أية عمليات حسابية على الثوابت sun ، sat ... إلى آخره . فلإيجاد اسم اليوم المعبر عن « باكر » أو « أمس » لا يجوز الجمع والطرح بل تستخدم لذلك الدوال **succ** ، **pred** كالمثال الآتى :

```
VAR yesterday, today, tomorrow:
               (mon,tues,wed,thurs,fri,sat,sun);
:

IF today = sun THEN tomorrow:= mon
               ELSE tomorrow:= succ (today)
:
               (أمس ) اليوم السابق —— اليوم التالى (باكر)
IF today = mon THEN yesterday:= sun
               ELSE yesterday:= pred (today)
```

شكل (٧ ــ ٢٥)

ويمكنك التأكد من أن الثوابت sat ، sun ،... تناظر أعداداً بطبع الكود المناظر لها باستخدام الدالة Ord كالآتى :

ord (mon) ← 0 تعطى القيمة

ord (sum) ← 6 تعطى القيمة

لا شك أن هذه الخاصية تسهّل على المبرمج مراجعة برنامجه حيث يصبح مكتوباً بلغة مفهومة كما أنها تسهل على القارىء متابعة ما يقرأ من عبارات باسكال . ولنر هذه العبارة الجميلة المعبرة عن قائمة من ألوان الطبيعة :

VAR colour : (red,green,blue,white);

إنها تعبّر عن نفسها .

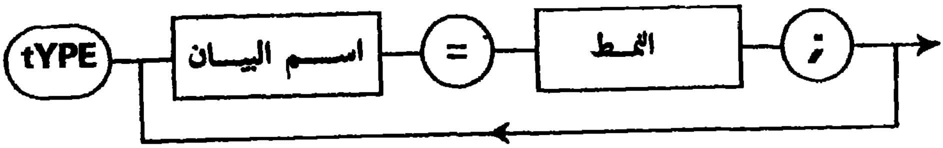
(٧ـــ٨ـــ٣) العبارة type :

فى الواقع أن الفئات الجزئية والقوائم ما هى إلاّ أنماط مبتكرة يمكن استخدام أسمائها تماماً كما تستخدم الأنماط القياسية والصورة الكاملة للأنماط المبتكرة تكتمل مع استخدام العبارة TYPE التى تعنى تعريف نمط جديد والمثال التالى يوضح استخدامها للتعريف بأيام الأسبوع وعدد الثوانى كأنماط جديدة تحت الإسمين day ، **minutetype** شكل (٧ ـــ ٢٦) .

```
TYPE minutetype = 1...59; ←—— فئة الدقائق
     day = (mon,tues,wed,thurs,fri,sat,sun);
VAR minutes : minutetype;
    today,tomorrow: day;         قائمة الأيام
```

شكل (٧ ـــ ٢٦)

وقاعدة استخدام العبارة type موضحة فى شكل (٧ ـــ ٢٧) بالرسم :



شكل (٧ ــ ٢٧)

● الثابت القياسى maxint :

لكل طراز من طرازات لغة باسكال ثابت قياسى يحمل الاسم maxint (اختصار maximum integer) بمعنى أكبر عدد صحيح موجب يمكن تمثيله فى الكومبيوتر . ولكى تكتشف قيمة هذا العدد فى جهازك الخاص اكتب هذا البرنامج :

البرنامج

```
begin
writeln('maxint is',maxint)
end.
```

```
Running
maxint is 32767
>
```

التنفيذ على IBM

شكل (٧ ــ ٢٨)

ومع الكومبيوتر الشخصى IBM يتوقع أن يكون هذا الرقم 32767 . ولهذا الثابت القياسى منافع مختلفة فى تعريف الأنماط المبتكرة . مثل تعريف نمط الأعداد غير السالبة (أى فئة الأعداد الموجبة والصفر) ، وتعريف نمط الأعداد الطبيعية (أى فئة الأعداد الموجبة) . والمثال التالى يوضح هذه التعريفات باستخدام الثابت القياسى maxint (شكل ٧ ــ ٢٩) :

```
TYPE nonnegative = 0..maxint;
     natural     = 1..maxint;
```

فئة الأعداد غير السالبة ← nonnegative

فئة الأعداد الطبيعية ← natural

شكل (٧ ــ ٢٩)

● **الفئات الجزئية من القوائم :**

إن الإعلان عن نمط من أنماط القوائم يتيح استخدام أسماء البيانات الواردة فى القائمة كثوابت . وبالتالى يجوز استخدامها فى الإعلان عن فئة جزئية جديدة .

والمثال التالى يقدم النمط day الذى يحتوى على قائمة بأيام الأسبوع ثم يستخدم القائمة ذاتها فى التعريف بالفئة الجزئية المعبرة عن أيام العمل (من الأحد إلى الخميس) بفرض أن أيام العطلات هى الجمعة والسبت ، وكذلك بالفئة الجزئية لأيام العطلات .

```
type day = (mon,tue,wed,thurs,fri,sat,sun);
workday = sun..thurs;   ←  أيام العمل
holiday = fri..sat;     ←  أيام العطلات
```

شكل (٧ ــ ٣٠)

●● ولعل المثال الأخير يوضح الفارق الأساسى فى المعنى بين القائمة والفئة الجزئية فكلاهما يعبر عن فئة ولكن الفئة الجزئية هى فئة متصلة أما القائمة فهى فئة غير متصلة . وعلى سبيل المثال فإن لاعبى كرة القدم ينتمون إلى فئة الفريق ، فإذا ميّزناهم بأسمائهم فإنهم يعتبرون **قائمة** وإذا ميّزناهم بأرقام مسلسلة فإنهم يصبحون **فئة جزئية** ! .

● **النمط البوليانى type boolean :**

يمكن تعريف النمط البوليانى القياسى بالصورة الآتية باعتباره نمط مبتكر :

TYPE boolean = (false,true);

وبذلك فإن الدالة **ord(false)** تعطى القيمة صفراً .

والدالة **ord(true)** تعطى القيمة 1 .

ولذلك فالنمط البوليانى ينتمى إلى أنماط القوائم .

● **الموجز :**

يمكن الإعلان عن الفئات الجزئية من الأنماط الآتية :

١ ــ الصحيح	integer
٢ ــ اللبنة	char
٣ ــ البوليانى (ولكن لا قيمة له)	boolean

كما يمكن الإعلان عن أنماط القوائم من أى نوع ما عدا النمط الصحيح real .

٧ ــ ٩) أدلة المصفوفات :

بعد أن عرضنا موضوع الفئات الجزئية والقوائم لعلنا نرى العلاقة بين المصفوفة وبين الفئات والقوائم واضحة الآن .

فعندما نعلن مصفوفة صحيحة مثل :

array [1..9] of integer

فإن أدلة هذه المصفوفة ما هى إلاّ الفئة الجزئية من الأعداد الصحيحة من 1 إلى 9 . كما رأينا أيضاً أن أدلة المصفوفة يجوز أن تكون فئة جزئية من اللبنات مثل :

array ['a'..'z'] of integers

وأدلة المصفوفات بصفة عامة يجوز أن تكون من أحد الأنواع الآتية :

١ ــ فئات جزئية من الأعداد الصحيحة .

٢ ــ فئات جزئية من اللبنات .

٣ ــ فئات جزئية من الثوابت المنطقية .

٤ ــ القوائم بأنواعها .

مثال (٧ ــ ١٠) :

وفى المثال التالى نرى أن دليل المصفوفة عبارة عن قائمة تحتوى على شهور السنة .

```
TYPE monthype = ( jan, feb, mar, apr,
                  may, jun, jul, aug,
                  sep, oct, nov, dec );
VAR  rainfall : ARRAY [ monthype ] OF real;
     thismonth,
     finalmonth : monthype;
     total : real;
BEGIN
     finalmonth := aug;
     total := 0.0;
     FOR thismonth := jan TO finalmonth DO
         total := total + rainfall [ thismonth ];
END.
```

أدلة المصفوفة

شكل (٧ ـ ٣١)

مثال (٧ ـ ١١) :

أما المثال التالى فهو يستخدم أدلة كفئة جزئية من الأعداد الصحيحة الممثلة للسنوات من 1919 حتى 1938 ، أما نمط المصفوفة نفسها فهو من الأنماط المبتكرة وهو عبارة عن فئة الأعداد غير السالبة (posint) أى التى تبدأ من الصفر وحتى maxint .

```
TYPE posint = 0..maxint;
     betweenwars = 1919..1938;
VAR  unemployed : array [betweenwars] OF posint;
     year       : betweenwars;
     posvar     : posint;
     intvar     : integer;
```

شكل (٧ ـ ٣٢)

فى هذا المثال يمكن الإشارة إلى أحد المتغيرات الدليلية بالمصفوفة كالآتى :

unemployed[year]

حيث يعبر المتغير year عن أحد السنوات فى الفئة [1919..1938] .

● **كما يجوز الإشارة إلى المتغير الدليل للمصفوفة باستخدام الدليل** (posvar) الذى تم تعريفه بأنه عدد صحيح موجب .

unemployed[posvar]

ولكن استخدام مثل هذا الدليل سيتطلب من البرنامج إجراء مراجعة أثناء التنفيذ للتأكد من أن قيمة هذا المتغير تقع فى نطاق الفئة **[1919.1938]** ، فهذه الفئة تمثل حدود الدليل المستخدم للمصفوفة .

وكذلك الحال إذا استخدمنا الدليل (intvar) الذى تم تعريفه كعدد صحيح .

والاستخدام الأمثل هو استخدام الدليل الأول (year) .

مثال (٧ ــ ١٢) :

وهذا مثال لمصفوفة تحتوى على أربعة مواسم تجارية تم الإعلان عنها كقائمة من أربعة عناصر (quarter) وأطلق عليها الاسم figures كنمط مبتكر .

```
TYPE quarter = (first, second, third, fourth);
     figures = ARRAY [quarter] OF integer;
     period  =1976..1980;
VAR  profits, losses, turnover : figures;
     fiveyearprofits : ARRAY [period] OF figures
     current : quarter;
```

(مصفوفة ذات بعد واحد)

(مصفوفة ذات بعدين)

شكل (٧ ــ ٣٣)

ونلاحظ أنه تم تعريف متغيرات ثلاثة كمصفوفات من نفس النمط . هذه المتغيرات هى :

الأرباح	**profits**
الخسائر	**losses**
جملة المبيعات	**turnover**

فإذا أردنا تمثيل أرباح الموسم التجارى ، فيمكننا الإشارة إليه بالمتغير الدليلى :

profits [second]

وجملة مبيعات الموسم الثالث هى :

turnover [third]

أما خسائر الموسم الحالى (current) فهى :

Losses [current]

●● وإذا دققنا النظر للإعلان عن المتغير fiveyearprofits لوجدنا أنه « **مصفوفة مصفوفة** » أى مصفوفة ذات بعدين . وهذه وجهة نظر أخرى للمصفوفة ذات البعدين حيث يمكن اعتبارها مكونة من مصفوفات ذات بعد واحد وهى الصفوف (أو الأعمدة) . لذلك يمكن الإشارة إلى أحد عناصر هذه المصفوفة كالآتى :

fiveyearprofits [1977,first]

وهو يمثل أرباح الموسم الأول من ١٩٧٧ .

كما يجوز التعبير عن نفس المتغير كالآتى :

fiveyearprofits [1977] [first]

٧ ـــ ١٠) المصفوفات المُحزّمة *Packed arrays* :

هذه نوعية من المصفوفات صممت لتشغل حيزاً أقل من سعة الذاكرة لكنها لا تتوفر بكل طرازات اللغة .

ويتم الإعلان عن هذه النوعية من المصفوفات بإضافة كلمة packed قبل كلمة array فى الإعلان عن المصفوفة .

وهذا الإعلان يتم فيه التعريف بمصفوفتين A1 ، A2 الأولى من النوع المحزّم والثانية من النوع العادى وكلاهما من النمط الحقيقى :

```
VAR     A1 : PACKED ARRAY [1..4] OF real;
        A2 :   ARRAY [1..4] OF real;
```

وفائدة المصفوفات المحزمة تظهر فى التعامل مع الحرفيات حيث أن لغة باسكال القياسية قد عرّفت الحرفيات (المتغيرات الحرفية) بأنها مصفوفة لبنات محزمة كالآتى :

```
PACKED ARRAY [1..n] OF char;
```

حيث n هو عدد اللبنات فى الحرف المقصود .

ومن خصائص عبارة الطباعة (write أو writeln) أنها تطبع هذه الحرفيات (راجع الباب الثالث) .

ويتم التعامل مع مصفوفة الحرفيات المحزمة بنفس الأسلوب الذى يتم به التعامل مع المصفوفات .

فيمكن التخصيص لمتغير حرفى كالآتى (بنفس عدد اللبنات) :

```
VAR name : PACKED ARRAY [1..10] OF char;
name := 'pascal    ';
```

كما يمكن مقارنة الحرفيات باستخدام المؤثرات العلاقية كالآتى :

```
IF name = 'pascal
   THEN...
IF name < 'pascal
   THEN...
```

وتتم المقارنات بين الحرفيات لبنة بلبنة من اليسار إلى اليمين وفقاً للكود آسكى (ASCII) أو الكود EBCDIC فى بعض الطرازات .

مثال (٧ ــ ١٣) على الكومبيوتر آى . بى . إم :

يوضح شكل (٧ ــ ٣٤) برنامجاً يستخدم مصفوفة اللبنات المحزّمة المكونة من ١٢ عنصراً .

ويبدأ البرنامج بقراءة الاسم من لوحة الأزرار ثم يقوم البرنامج باختبار الاسم الذى تمت قراءته إذا كان هو الاسم « عبد الفتاح » وفى حالة إذا ما تحقق الشرط فإنه يعطى الرسالة :

ok...name found

وإلاّ فإنه يطبع الاسم المدخل كما هو .

ونلاحظ مع العبارة read أنه يمكن إدخال عدد من اللبنات أقل من أبعاد المصفوفة . وهذا لا يجوز عند استخدام التخصيص .

البرنامج

```
program demo(input,output);
var name:packed array[1..12] of char;

begin
readln(name);
if name = abdel-fattah  then
     writeln('ok..name found.')
      else writeln(name)
end.
```

التنفيذ

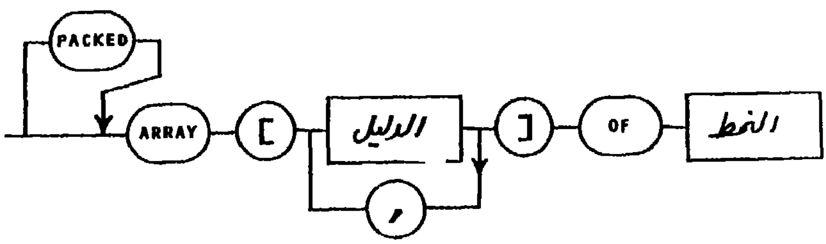
```
Running
ahmad      ← الاسم المدخل: "أحمد"
ahmad      ← لم يتحقق الشرط
:

Running
abdel-fattah     ← الاسم المدخل "عبدالفتاح"
ok..name found.  ← تحقق الشرط

:
```

شكل (٧ ــ ٣٤)

والشكل الآتى يوضح قاعدة تكوين المصفوفات والمصفوفات المحزّمة بالرسم :



شكل (٧ ــ ٣٥)

(٧ ــ ١١) الحرفيات فى طرازات باسكال string :

كما نرى أن الحرفيات فى اللغة القياسية كانت مقيّدة ! ولكن بعض الطرازات قد أدخلت الحرفيات بمعناها الدقيق كما فى اللغات الأخرى .

وقد تم ذلك بإدخال نمط جديد هو **النمط الحرفى string** بحيث يمكننا أن نعلن عن متغيّر حرفى بالصورة الآتية :

```
VAR name : string;
```

وهذا المتغير الحرفى يسمح باستقبال ثوابت حرفية يصل طولها إلى ثمانين لبنة (فى الطراز UCSD) .

وعلاوة على ذلك فإنه يمكن الإعلان عن أى طول متوقع للحرفى يزيد عن الثمانين كالآتى :

```
string [n]
```

حيث n : عدد صحيح لا تزيد قيمته عن ٢٥٥ .

●● **ملاحظة** : هذه الصيغة المسموح بها فى الكومبيوتر IBM والطرازات المتوافقة معه أما الصيغة الأولى فغير متوفرة) .

ويتم الإعلان كالمثال الآتى :

```
VAR carreg : string [7];
    printline : string [132];
```

وعند التخصيص للمتغير الحرفى يجوز أن نخصص له ثابتاً حرفياً بطول أقل أو يساوى الطول المعلن عنه كالآتى :

```
carreg := 'abc123x'
```

أو

```
carreg := 'aa1'
```

كما يجوز فهرسة الحرفى (على اعتبار أنه مصفوفة) وفى هذه الحالة يمكن الإشارة إلى أى لبنة فيه كأنها متغيّر دليلى له ترتيب معيّن كالمثال الآتى :

```
VAR name : string;
    ch   : char;
name := 'kenneth';
ch := name [6]; الحرف السادس
```

شكل (٧ ــ ٣٦)

فالمتغير اللبنة ch قد أصبح محتوياً على الحرف t بعد عملية التخصيص حيث أن الحرف t هو الحرف السادس فى مصفوفة المتغير الحرفى name .

(٧ ــ ١٢) دوال الحرفيات *string functions* :

إن الطرازات التى يتوفر بها النمط string تمدنا فى نفس الوقت بكل الوسائل اللازمة لمعالجة الحرفيات كما فى لغة بيسك .

وهذه الوسائل عبارة عن دوال وبرامج متوّفرة ضمن مترجم اللغة تماماً كالدوال القياسية .

(١) قياس طول الكلمة length :

صيغة هذه الدالة :

Length (string)

وهى تستخدم لإيجاد عدد لبنات الحرفى (string) الذى بين القوسين ، ونتيجة هذه الدالة عدد صحيح يجوز تخصيصه لأى متغير صحيح .

مثال (٧ ــ ١٤) :

البرنامج التالى يستقبل اسماً ما name ويخزّنه فى متغير حرفى يتسع لعشرين لبنة ثم يقيس طول الاسم (عدد لبناته) ويخصصه للمتغير الصحيح l .

البرنامج

```
program length(input,output);
var name:string[20];
     l:integer;

begin

readln(name);
l:=length(name);
writeln(name,' of length ',l);

end.
```

قياس طول الكلمة

التنفيذ

```
Running
ibrahim soliman  <-  الاسم المدخل
ibrahim soliman of length 15  <-  طول الاسم

>

Running
sally o. al-husseiny  <-  الاسم المدخل
sally o. al-husseiny of length 20  <-  طول الاسم

>
```

شكل (٧ ــ ٣٨)

(٢) ترتيب لبنة داخل حرفى POS :

تعطى هذه الدالة عدداً صحيحاً يعبر عن ترتيب لبنة معنية أو أكثر (حرفى جزئى) بداخل حرفى وإذا كان الحرفى الجزئى متكرراً أكثر من مرة بداخل هذا الحرفى فإن هذه الدالة تعطى ترتيباً **أول حدوث** للحرفى الجزئى .

وصيغة الدالة هى :

pos (substring, string)

حيث :

substring الحرفى الجزئى المطلوب إيجاد موضعه .

string الحرفى المعلوم .

مثال (٧ ــ ١٥) :

هذه الشريحة من البرنامج تبحث عن ترتيب الكلمة and فى المتغير الحرفى authors وتخصص النتيجة للمتغير الصحيح I وعند طباعة قيمة المتغير I سوف تكون 8 لأن بداية الكلمة and تقع فى الخانة الثامنة .

```
VAR l : integer;
    authors : string [20];
authors := 'jensen and wirth';
l := pos ('and', authors);
```

شكل (٧ ــ٣٩)

وإذا فشلت عملية البحث عن الحرفى الجزئى فإن الدالة تعطى القيمة صفراً . ولنر هذا المثال :

(مثال (٧ — ١٦) :

يوضح الشكل التالى برنامجاً يستقبل اسماً ويخزنه فى الحرف name ثم يستقبل لبنة واحدة ليبحث عن وجودها فى الحرف name ثم يطبع على الشاشة موضع هذه اللبنة من الاسم . فإذا كانت اللبنة غير موجودة أعطى القيمة صفراً .

```
program position(input,output);                    البرنامج

var name:string[255];
     l:integer;
     letter:char;                      البحث عن حرف فى اسم

begin
writeln('please enter a name...');
readln(name);
writeln('what is character you''re looking for?');
readln(letter);
l:=pos(letter,name);
writeln(letter,' is in the position no:',l);

end.

>
                                                      التنفيذ
Running
please enter a name...
mahmood abbas  <----------  الاسم المدخل
what is character you're looking for?
b <---- الحرف المراد البحث عنه
b is in the position no:10
                                   الترتيب
>

Running
please enter a name...
aly abdel-gawwad <---------  الاسم المدخل
what is character you're looking for?
z <---- الحرف المراد إيجاد موضعه
z is in the position no:0 <---- الحرف غير موجود

>
```

شكل (٧ — ٤٠)

وبالطبع يمكن تعديل البرنامج بحيث يخبرنا برسالة مفهومة : إذا ما كانت اللبنة المراد إيجاد موضعها موجودة أصلاً فى الحرف أم لا .

وهذا التعديل نراه فى البرنامج شكل (٧ — ٤١)

* * *

البرنامج

البحث عن حرف في كلمة

```
program position2(input,output);
var name:string[255];
     l:integer;
     letter:char;

begin
writeln('please enter a name...');
readln(name);
writeln('what is character you''re looking for?');
readln(letter);
l:=pos(letter,name);
if l=0 then
writeln('the character doesn' t exist')
else
writeln(letter,' is in the position no: ',l);

end.
```

التعديل المطلوب ← (if l=0 then / writeln(...) / else)

التنفيذ

```
please enter a name...
MEDHAT ASSAR          ← الاسم
what is character you re looking for?
s                     ← الحرف المراد البحث عنه
the character doesn't exist   ← النتيجة .. الحرف غير موجود

>

Running
please enter a name...
medhat assar          ← الاسم
what is character you're looking for?
s                     ← الحرف المراد البحث عنه
s is in the position no:9   ← النتيجة

/

Running
please enter a name...
abdel-salam shamah    ← الاسم
what is character you're looking for?
r                     ← الحرف
the character doesn't exist   ← غير موجود
```

قارن بين هذين التنفيذين

شكل (٧ ـ ٤١)

(٣) وصل الكلمات concat :

تقوم هذه الدالة بوصل الحرفيات بعضها ببعض وصيغتها كالآتى :

concat (string, string, ..string)

ونتيجة الدالة هى عبارة عن حرفى مكون من الحرفيات الموجودة بداخل القوس ، والمفصولة عن بعضها البعض بفاصلة .

مثال (٧ ـــ ١٧) :

يوضح شكل (٧ ـــ ٤٢) التنفيذ المتوقع للبرنامج الذى سنعرضه حيث يستقبل ثلاثة أسماء هى : اسم الشخص واسم الأب (أو الاسم الأوسط) واسم العائلة ثم يطبع الاسم كاملاً مع اختصار الاسم الأوسط إلى حرف واحد يعقبه نقطة .

التنفيذ المتوقع للبرنامج

```
Running
please enter first name...
SALLY  <----   الاسم الأول
please enter the middle name...
OSSAMA <----   اسم الأب
please enter the family name
ABOLROUS <---- اسم العائلة
the complete name is :SALLY O. ABOLROUS

>
```

البرنامج يطبع الاسم كاملاً
مع اختصار اسم الأب

شكل (٧ ـــ ٤٢)

أما البرنامج فيمكن إنشاؤه كما هو موضح فى شكل (٧ ـــ ٤٣) حيث يستقبل الأسماء الثلاثة a,b,c وكلها متغيرات حرفية تمثل الاسم واسم الأب

واسم العائلة بالترتيب .

ثم يلى ذلك اختصار الاسم الأوسط إلى حرف واحد وذلك بتخصيص الحرف الأول منه **b[1]** للمتغير l .

يلى ذلك استخدام دالة الوصل لكى تصل بين الحرفيات الثلاثة a,l,c علاوة على المسافات الخالية بين الأسماء .

البرنامج ١

وصل الحرفيات

```
program completename(input,output);

var a,b,c,d:string[20];
    l:string[1];
begin
writeln('please enter first name...');
readln(a);
writeln('please enter the middle name...');
readln(b);
writeln('please enter the family name');
readln(c);

l:=b[1];          { ← اختصار الاسم الأوسط }
d:=concat(a,' ',l,'. ',c);   { ← الوصل }

writeln('the complete name is :',d);

end.
```

شكل (٧ ـــ ٤٣)

أما البرنامج شكل (٧ ـــ ٤٤) فهو يعرض نفس البرنامج مع الاستغناء عن المتغير الوسيط ا والاستعاضة عنه بتعديل سعة المتغيّر (b) المخصص لاستقبال الاسم الأوسط لتكون لبنة واحدة فقط أى :

b:string[1]

فى هذه الحالة ... عندما يُستقبل الاسم الأوسط أثناء تشغيل البرنامج لن يُختزن منه فى المتغير (b) سوى حرف واحد فقط .

```
program completename(input,output);

var a,c,d:string[20];
    b:string[1];
begin
writeln('please enter first name...');
readln(a);
writeln('please enter the middle name...');
readln(b);
writeln('please enter the family name');
readln(c);

d:=concat(a,' ',b,'. ',c);

writeln('the complete name is :',d);

end.
```

البرنامج ٢

متغير يتسع للبنة واحدة

وصل الحرفيات

الوصل

شكل (٧ ـــ ٤٤)

(٤) نسخ جزء من كلمة copy :

تقوم هذه الدالة بنسخ جزء من حرفى معلوم ، بمعلومية بداية الجزء المطلوب وعدد لبناته (أو طوله) .

وصيغة الدالة كالآتى :

copy (string, start, length)

حيث : string الحرفى المعلوم .

start بداية الحرفى الجزئى المطلوب نسخه .

length طول الحرفى الجزئى (عدد اللبنات) .

والبرنامج التالى يطبع اسم العائلة « عاشور » من الاسم الكامل « مصطفى عاشور » ويخصصه للمتغير lastname ثم يطبعه .

البرنامج

```
program copy(input,output):
var name:string[30];
    lastname:string[10];
begin
name:='MOUSTAFA ASHOOR';
lastname:= opy(name,10,6);
   writeln(lastname)

end.
```

الاسم الكامل

بدءا من الخانة رقم عشرة

وبطول قدره ست لبنات

التنفيذ

```
Running
ASHOOR
```

اسم العائلة

شكل (٧ ـ ٤٥)

(٥) إدماج كلم فى جملة insert :

وهذه خاصية معروفة من خصائص برامج التحرير (editors) أو معالجة

الكلمات (word processing) حيث يمكن « حشر » كلمة أو مجموعة كلمات بداخل نص معيّن .

والدالة التى تؤدى هذه الوظيفة هى insert وصيغتها كالآتى :

insert(string1 ,string2 position)

حيث : string1 الحرفى الجزئى (الكلمة) المراد إدماجها .

string2 الحرفى المعلوم (الجملة) .

position الخانة التى يبدأ عندها الإدماج .

ونتيجة هذه الدالة لا يجوز تخصيصها لأى متغيّر فهى لا تُنتج أية قيمة ولكنها تقوم بتغيير محتويات الحرفى string1 .

مثال (٧ ــ ١٨) :

البرنامج

```
program insert(input,output);
var names:string[30];
    l:string[40];
begin
names:='HAZEM SALLY';
insert('and  ',names,7);
   writeln(names)
end.
```

الحرفى المعلوم

الكلمة المراد إدماجها

بدءاً من الخانة رقم 7

التنفيذ

```
Running
HAZEM and  SALLY
```

النتيجة

شكل (٧ ــ ٤٦)

فى هذا المثال كان الحرف المعلوم هو "HAZEM SALLY" وقد تم إدماج كلمة and فى الخانة السابعة فأصبح الحرف الجديد هو :

"HAZEM AND SALLY"

(٦) حذف كلمة من جملة delete :

هذه الدالة تؤدى العملية العكسية للدالة insert وصيغتها كالآتى :

delete(string ,position, length)

حيث : string الحرف المعلوم المراد حذف جزء منه .
position الخانة التى يبدأ عندها الحذف .
length طول الجزء المراد حذفه باللبنات .

مثال (٧ ــ ١٩) :

فى هذا المثال نقدم الحالة العكسية للمثال السابق حيث نحذف الجزء "and HAZEM" من الحرف "SALLY and HAZEM" فيصبح SALLY بعد التعديل .

```
program delete(input,output);
var names:string[30];

begin
names:='SALLY and HAZEM';
delete(names,6,10);
 writeln(names)

end.


>

Running
SALLY

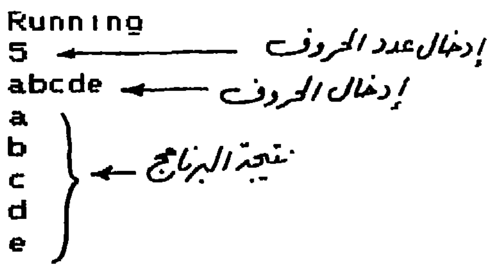
>
```

البرنامج
الحرف المعلوم
يبدأ من الخانة رقم ٦
وبطول قدره ١٠ لبنات
اطبع الحرف بعد التعديل
التنفيذ
النتيجة

شكل (٧ ــ ٤٧)

■ تمرينات الباب السابع :

س (٧ — ١) اكتب برنامجاً يستقبل مصفوفة من الحروف الأبجدية (بحد أقصى عشرون حرفاً) ، ويطبعها بالصورة الآتية :



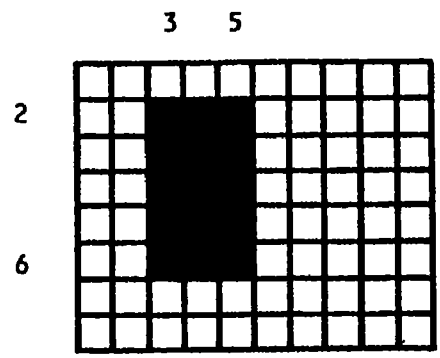
شكل (٧ — ٤٨)

س (٧ — ٢) اكتب برنامجاً لتحليل مبيعات الأقسام لشركة من الشركات باستخدام الرسم البيانى مستخدماً مقياس رسم **(10)** إذا كانت البيانات تستقبل فى الكومبيوتر بالصورة الآتية :

الرقم المُدخل	المعنى
5	**عملية بيع واحدة للقسم رقم 5**
3	**عملية بيع واحدة للقسم رقم 3**
8	**عملية بيع واحدة للقسم رقم 8**
.	
.	
.	
1 –	**نهاية البيانات**

ملاحظة : تستخدم الشركة ٨ أقسام للبيع .

س (٧ ــ ٣) كتطوير للمثال رقم (٧ ــ ٨) الذى يعالج تعداد السكان ، المطلوب طباعة عدد السكان فى منطقة جزئية من المدينة والتى يتم تعريفها بأربعة أرقام مثل 5 3 6 2 (أنظر شكل ٧ ــ ٢٠) أى المنطقة المحصورة بين الصفيّن 2,6 والعمودين 3,5 وقد تم تظليل المنطقة المقصورة بالرسم .



شكل (٧ ــ ٤٩)

س (٧ ــ ٤) اكتب برنامجاً يقرأ قطعة من نص تنتهى بالعلامة (*) ويقوم البرنامج بإيجاد الحرف ذى أعلى معدّل تكرار فى النص .

س (٧ ــ ٥) اكتب برنامجاً يقرأ عدداً صحيحاً (n) متبوعاً بكلمة تحتوى على عدد (n) من الحروف . ويقرر البرنامج إذا ما كانت الكلمة تقرأ من اليمين كما من اليسار (أى مثل كلمة ROTOR) .

س (٧ ــ ٦) ينقسم الأسبوع الدراسى لأحد الطلبة إلى خمسة أيام دراسية يحتوى كل منهم على ٦ فترات دراسية (محاضرات أو معامل) ويجب على الطالب

أن يحضر ٢٠ فترة خلال الأسبوع الواحد . والبرنامج المطلوب يقوم بقراءة قائمة بمواعيد المحاضرات وأماكنها حيث يتم تمثيل كل محاضرة برقمين يعبر أحدهما عن اليوم (5..1) والآخر يعبر عن الفترة (6..1) ويلى هذين الرقمين رقم ثالث يدل على قاعة المحاضرات أو المعمل .

والمطلوب من البرنامج أن يطبع الجدول الزمنى للمحاضرات بالصورة الآتية :

الفترة الدراسية ← period

أيام الأسبوع / رقم القاعة	1	2	3	4	5	6
mon	3	1	2	9		
tue	6	6	5	7	6	
wed		2	1			
thu	5	2	5		8	7
fri	3	2	1	3		

شكل (٧ ــ ٥٠)

س (٧ ــ ٧) اكتب إعلاناً عن نمط مبتكر يمثل « العام » فى القرن الحالى على أن يكون النمط صحيحاً وممثلاً بأربعة أرقام مثل ١٩٨٨ .

س (٧ ــ ٨) تنقسم الدراسة فى أحد الكليات إلى ثلاثة ترمات (terms) الترم الأول first ، والثانى second ، والصيفى summer . اكتب إعلاناً عن النمط term وعن متغيرين هما thisterm (الترم الحالى) و nexterm (الترم التالى) .

ثم اكتب عبارة (عبارات) لتخصيص قيمة ما للمتغير nexterm إذا أعطيت قيمة المتغير thisterm .

س (٧ ـــ ٩) اكتب إعلاناً عن النمط ''matrix'' الذى يحتوى على ٣ صفوف وخمسة أعمدة من الأعداد الحقيقية .

ثم أعلن عن ثلاث مصفوفات أخرى c,b,a . وبفرض أن هناك قيماً تم تخصيصها لكل من b,a اكتب العبارات اللازمة لتوليد قيم المصفوفة c بحيث يكون كل عنصر منها يمثل مجموع العنصرين المناظرين فى كل من المصفوفتين b,a .

س (٧ ـــ ١٠) اكتب برنامجاً بلغة باسكال لحساب عدد مرات حدوث الحروف 'a' ، 'z' فى ملف من اللبنات . أهمل علامات نهاية السطر واللبنات غير الأبجدية .

ثم اطبع عدد مرات تكرار كل حرف .



الباب الثامن

البرامج الفرعية والدوال

Procedures & Functions

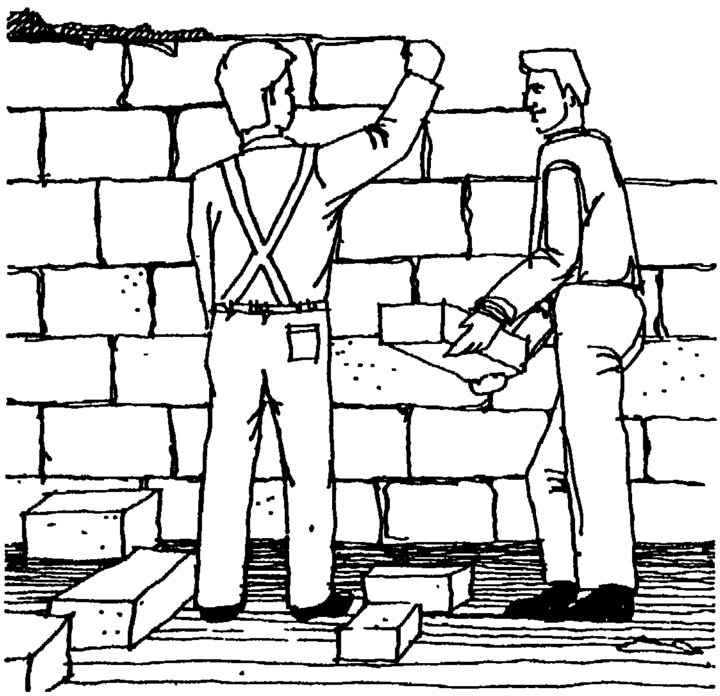
مفتتح

إذا استعرضنا منطق أحد البرامج الكبيرة سوف نجد أنه يهدف إلى إنجاز عمل ما وهذا يحدد الصفة الأساسية المميزة للبرنامج . ولكن مهما كان البرنامج متفرداً فى هذا الهدف فهو يحتوى فى تفصيلاته على عمليات أخرى متكررة . وهذه العمليات المتكررة قد تكون عمليات شائعة الاستخدام مثل حساب الجذر التربيعى وجيب الزاوية أو حتى بعض العمليات الأكثر تعقيداً مثل حل معادلة رياضية أو إحصائية . وقد تكون العملية المتكررة مبتكرة ومتفرّدة أيضاً ينشئها المبرمج وفقاً لاحتياجاته .

وسواء كانت العملية المتكررة شائعة أو متميّزة فلا معنى لتكرار برمجتها داخل البرنامج كلما احتجنا إليها . فمن المفضّل فى هذه الحالة أن نُبرْمج مرة واحدة وتستدعى عند الحاجة إليها .

وقد رأينا فى الباب الثانى أن لغة باسكال تمدنا بمجموعة من البرامج الجاهزة فى مكتبة الدوال القياسية لكى نستخدمها فى العمليات شائعة الاستخدام . أما إذا كانت العملية المتكررة مبتكرة فيمكن فى هذه الحالة تقسيم البرنامج الكبير إلى وحدات منفصلة : برنامج رئيسى وبرامج فرعيّة ، حيث يختص البرنامج الرئيسى بمنطق البرنامج الأصلى وتختص البرامج الفرعية بالعمليات الجانبية المتكررة .

وفى هذا الباب نتعرض لكيفية بناء البرامج باستخدام الوحدات المساعدة : البرامج الفرعية والدوال المبتكرة .



بناء البرنامج من برامج فرعيّة

شكل (٨ ــ ١)

(٨ ــ ١) البرنامج الفرعى والبرنامج الرئيسى

procedure & program

إذا كنا بصدد تصميم برنامج لحساب أجور الموظفين بمؤسسة ما ، فمن البديهى أن العمليات التى يتضمنها البرنامج سوف تحتوى على المعالجات الآتية :

١ ــ تجميع أجر ساعات العمل العادى لكل موظف .
٢ ــ تجميع أجر ساعات العمل الإضافية لكل موظف .
٣ ــ خصم ضريبة الدخل العام .

لا بأس عندئذ من إنشاء ثلاثة برامج صغيرة يختص كل منها بعمل من هذه الأعمال ولتكن أسماؤها بالترتيب :

addnormalhourspay
addovertimehourspay
deductincometax

وعند بناء البرنامج الرئيسى الذى يتولى حساب الأجور لن نحتاج أن نضمنّه أية خطوات لحساب الساعات الأصلية أو الإضافية إلى آخره بل كل ما نحتاجه أن نكتب اسم أحد هذه البرامج الفرعية بداخل البرنامج الرئيسى ، فيتولى البرنامج الرئيسى استدعاءها وتنفيذها . عندئذ يصير البرنامج الفرعى كأنه عبارة من عبارات باسكال وعادة يكتب البرنامج الفرعى (procedure) بداخل البرنامج الرئيسى (program) بعد إعلان المتغيرات مباشرة ، أى قبل بداية البرنامج الرئيسى (Begin) .

والشكل (٨ ــ ٢) يوضح علاقة البرنامج الفرعى B بالبرنامج الرئيسى A كشكل عام .

```
program  A(input,output);
VAR.....
     .....                البرنامج الرئيسى A

procedure  B;
VAR.....
begin
.....                                        البرنامج الفرعى B
end;

begin
...
B      ←            استخدام البرنامج الفرعى
                    كخطوة من خطوات المعالجة
...
end
```

شكل (٨ ـــ ٢)

مثال (٨ ـــ ١) :

ينتج مصنع للأدوات الميكانيكية « محاور » ذات أقطار مختلفة من ماكينتى إنتاج . وفى خط الإنتاج تؤخذ عينة مكوّنة من عشرة محاور من الماكينة الأولى ويتم حساب متوسط أقطارها ثم تؤخذ عينة من الماكينة الثانية ويحسب متوسط أقطارها . وفى النهاية يتم حساب المتوسط العام للعشرين .

والبرنامج التالى يقوم بهذه العملية وهو يتكون من برنامج رئيسى **overage20** وبرنامج فرعى **overage10** .

وكما هو واضح من أسماء البرامج فالبرنامج الفرعى يختص بحساب المتوسط لعشرة قيم ، أما الرئيسى فهو يحسب المتوسط النهائى لمجموعتين من العينّات مكونة من ٢٠ قطعة . شكل (٨ — ٣) :

```
program average20(input,output);
var overallsum : real;
   procedure average10;
   var count : integer;  next, sum10 : real;
   begin
      sum10 := 0;
      for count := 1 to 10 do
      begin
         read(next);
         sum10 := sum10 + next
      end;
      writeln('average of 10 readings :', sum10/10);
      overallsum := overallsum + sum10
   end;
begin
   overallsum := 0;
   average10;
   average10;
   writeln('overall average :', overallsum/20)
end.
```

البرنامج الرئيسى

متغير عام

متغير محلى

البرنامج الفرعى

بداية التنفيذ

شكل (٨ — ٣)

وكما نرى فإن البرنامج الرئيسى يبدأ بالفعل بعد البرنامج الفرعى أى أن تنفيذ البرنامج يبدأ بالعبارة :

overallsum: = 0

وهى العبارة الأولى بالبرنامج الرئيسى . يلى ذلك العبارة الثانية :

average 10

وهذه العبارة هى نفسها اسم البرنامج الفرعى وينتج عنها استدعاء وتنفيذ البرنامج الفرعى لذلك فهى تسمى عبارة نداء **(procedure call)** . والعبارة الثالثة هى أيضاً عبارة نداء لنفس البرنامج الفرعى . أما العبارة الأخيرة فهى تحسب المتوسط الكلى للعينات وتطبعه بالتعبير :

overallsum/20

ولنلق نظرة أكثر قرباً على متغيّرات البرنامج :

● **المتغيرات العامة** **Global variables**

فى فقرة الإعلانات بالبرنامج الرئيسى نلتقى بالمتغير overallsum وهذا المتغير يعتبر **متغيراً عاماً (global variable)** بمعنى أنه سيجوز استخدامه فى أى جزء من البرنامج .

● **المتغيرات المحليّة** **Local variables :**

أما فى بداية البرنامج الفرعى فنرى إعلاناً عن المتغيرات count, next, sum10 وهذه النوعية من المتغيرات لا يجوز أن تستخدم إلاّ فى البرنامج الفرعى . ولذلك يطلق عليها **المتغيرات المحليّة (local variables)** .

والسبب فى ذلك أن هذه المتغيرات تحتل خانات الذاكرة فقط أثناء استدعاء البرنامج الفرعى الذى يستخدمها فإذا انتهى تنفيذ البرنامج الفرعى يتم مسح هذه الخانات من الذاكرة توطئة لا ستخدامها مع بيانات البرنامج الرئيسى .

ولذلك فإن على المبرمج أن يقرر قبل كتابة البرنامج : أى المتغيّرات سوف يحتاج إليها أثناء البرنامج الرئيسى حتى يعلنها كمتغيرات عامة أما المتغيرات التى

تخدم البرنامج الفرعى فيكتفى بإعلانها بداخل البرنامج الفرعى فقط كمتغيرات محليّة . وهذا الأسلوب يجنّب استهلاك الذاكرة بلا داع .

ما فائدة البرنامج الفرعى ؟ .

لعلنا نرى فى هذا المثال أننا كتبنا البرنامج الفرعى مرة واحدة واستدعيناه مرتين ، وفى بعض التطبيقات قد نحتاج استدعاء البرنامج الفرعى عشرات المرّات . لذلك فقد أغنانا وجود البرنامج الفرعى عن تكرار كتابته عدة مرّات فى نفس البرنامج .

هذا فضلاً عن أنه يجعل البرنامج سهل القراءة على الإنسان فالبرنامج بهذه الصورة يتكون من أربع عبارات فقط سهلة القراءة سهلة الفهم .

(٨ ـــ ٢) *البارامترات* ***Parameters :***

فى المثال السابق رأينا أن البرنامج الفرعى يقوم بتنفيذ عملية محدّدة فى كل مرة ، يستقبل قيم نفس المتغيرات ، يعالجها ، ويطبع النتائج . ولكن هناك وسيلة تجعل البرنامج الفرعى أكثر مرونة بحيث يمكنه تنفيذ عمليات مختلفة أو التعامل مع متغيرات مختلفة . يتم ذلك بإمداد البرنامج الفرعى ببارامتر parameter . ولنضرب مثلاً .

مثال (٨ ـــ ٢) :

لنفرض أننا نريد كتابة برنامج لطبع فاتورة الكهرباء حيث يستقبل عددين يمثلان القراءة الحالية والقراءة السابقة (يحتوى كل عدد على أربعة أرقام) ثم يطبع القراءتين والمبلغ المطلوب . هذا المنطق قد يكون كالآتى :

١ ـــ اقرأ القراءة السابقة (previous) والقراءة الحالية (present) .

٢ ـــ اطبع العناوين « القراءة السابقة » ، « القراءة الحالية » ، « المبلغ المطلوب .

٣ ـــ اطبع القراءة السابقة .

٤ ـــ اطبع القراءة الحالية .

٥ ـــ اطبع المبلغ المطلوب .

ولأن الأعداد المكتوبة فى فاتورة الكهرباء عادة ما تكون رباعية الأرقام (أى أن العدد 45 على سبيل المثال يكتب على الصورة 0045) فلعله من المناسب أن نطلق الاسم **"write4digit"** على البرنامج الفرعى الذى يتولى كتابة قراءة العدّاد .

فلنكتب الآن البرنامج الفرعى الذى يمكن استخدامه مرّة لطباعة القراءة الحالية ومرة أخرى لطباعة القراءة السابقة.. وشكل (٨ ـــ ٤) يوضح صورة ممكنة لهذا البرنامج .

```
procedure write4digits(meterreading : integer);
begin
   write('        ');
   if meterreading < 10 then write('000')
   else if meterreading <  100 then write('00')
   else if meterreading < 1000 then write('0');
   write(meterreading) ,
end;
```

البارامتر

شكل (٨ ـــ ٤)

إن اسم البرنامج الفرعى ـــ كما اتفقنا عليه ـــ هو : "write4digit"
ولكن كلمات جديدة ظهرت ما بين القوسين عقب الاسم مباشرة وهى :

(meterreading:integer);

أما الاسم **meterreading** فيسمى « البارامتر » (وهو من النوع الصحيح فى هذه الحالة وفقاً للإعلان integer) . وعندما يستدعى البرنامج

الفرعى فإن هذا البارامتر يأخذ قيمة جديدة وهى القيمة المطلوب استخدامها فى البرنامج الفرعى بالفعل . فإذا استدعينا هذا البرنامج الفرعى بعبارة مثل :

write4digits(prvious)

فإن البارامتر meterreading يُستبدل بالمتغير previous وعليه فإن البرنامج الفرعى يطبع قيمة القراءة السابقة .

وأما إذا استدعيناه بالعبارة :

write4digits(present)

فإن المتغير present يستبدل البارامتر meterreding ويقوم البرنامج الفرعى بطبع قيمة القراءة الحالية .

وعادة ما يسمى المتغير present (أو المتغير previous) الذى يستخدم فى الاستدعاء **بالبارامتر الحقيقى** أما البارامتر بالبرنامج الفرعى فقد يسمى بالبارامتر الهيكلى .

ويجوز أن يحل محل البارامتر الهيكلى أى عدد صحيح (ثابت صحيح أو تعبير صحيح) إذ أنه ليس بالضرورة أن يكون متغيراً ، كالمثال الآتى :

write4digits(256 + 17)

فى هذه الحالة يكون الخرج هو العدد 0273 .

وشكل (٨ — ٥) يوضح الخطوات الأساسية من البرنامج الرئيسى التى تستدعى بها البرنامج الفرعى لطباعة فاتورة الكهرباء .

```
read(previous, present);   .. للقراءة السابقة
writeln('prev reading   pres reading    total due');
write4digits(previous);  write('   ');
write4digits(present);   .. للقراءة الحالية
writeln('£':8, (present - previous)*unitprice :5:2)
```

شكل (٨ — ٥)

أما شكل (٨ ـ ٦) فيوضح الشكل النهائى للبرنامج الرئيسى متضمناً البرنامج الفرعى write4digit .

```
program electricitybill(input,output);

const unitprice = 0.034;

var previous, present : integer;

   procedure write4digits(meterreading : integer);
   begin
      write('       ');
      if meterreading < 10 then write('000')
      else if meterreading <  100 then write('00')
      else if meterreading < 1000 then write('0');
      write(meterreading)
   end;

begin
   read(previous, present);
   writeln('prev reading   pres reading    total due');
   write4digits(previous);  write('   ');
   write4digits(present);
   writeln('£':8, (present - previous)*unitprice :5:2)
end.
```

البرنامج الرئيسى

البارامتر الهيكلى

البرنامج الفرعى

البارامترات الحقيقية

شكل (٨ ـ ٦)

(٨ ـ ٣) استخدام أكثر من بارامتر :

لا مانع من استخدام أكثر من بارامتر هيكلى فى نفس البرنامج الفرعى ولا بأس من استخدام بارامترات من أنماط مختلفة .

ولنلق نظرة على البرنامج الفرعى شكل (٨ ـ ٧) .

```
procedure writechar(ch : char;  nooftimes : integer);
var count : integer;
begin
   for count:= 1 to nooftimes do write(ch);
   writeln
end;
```

البارامترات

اللبنة المراد طبعها

عدد مرات التكرار

شكل (٨ — ٧)

هذا البرنامج الفرعي يستقبل عدداً صحيحاً ولبنة ، ثم يقوم بطباعة اللبنة عدداً من المرات يساوى العدد الصحيح الذى تم استقباله . فإذا أردنا استدعاء هذا البرنامج الفرعي فلنكتب عبارة مثل :

writechar('*', 5)

عند تنفيذ هذا الاستدعاء سوف يقوم البرنامج الفرعي بطباعة خمسة نجوم متجاورة حيث أن اللبنة (*) سوف تحل محل البارامتر ch والعدد 5 سوف يحل محل البارامتر nooftimes . والنقطة الجديرة بالملاحظة هنا هى أن البارامترات الهيكلية ظهرت مفصولة **بفاصلة منقوطة** فى البرنامج الفرعى **لأنها تنتمى إلى أنماط مختلفة** .

أما فى عبارة الاستدعاء فقد ظهرت البارامترات الحقيقية مفصولة **بفاصلة عادية** .

وكما ذكرنا من قبل يمكن أن تكون البارامترات الحقيقية المستخدمة فى استدعاء البرنامج الفرعى — تعبيرات أو متغيّرات من نفس النوع . فإذا كان لدينا متغير لبنة اسمه **symbol** (بمعنى رمز) ومتغير صحيح اسمه **noofsymbols** (بمعنى عدد الرموز) ، فيمكننا كتابة العبارات التالية على سبيل المثال (٨ — ٨) .

```
symbol := 'a';  noofsymbols := 5;
writechar(symbol, noofsymbols);
writechar(succ(succ(symbol)), 2*noofsymbols + 1)
```

شكل (٨ ــ ٨)

سينتج عن هذه العبارات طباعة الحرف a خمس مرات نتيجة لتنفيذ العبارة الثانية ، والحرف c إحدى عشر مرة نتيجة لتنفيذ العبارة الثالثة (تذّكر معنى الدالة **succ**) .

أى أن الخرج سيكون :

aaaaa

ccccccccccc

ماذا لو كانت البارامترات الهيكلية من نفس النوع (النمط) ؟ .

إنها فى هذه الحال يمكن أن تظهر **مفصولة بفاصلة عادية** . والبرنامج الفرعى التالى (شكل ٨ ــ ٩) يوضح مثالاً لاستخدام مجموعات من البارامترات متشابهة وغير متشابهة .

```
procedure demo(i,j :integer;  a,b,c,d :char;  x,y,z :real);
begin
  •
  •
  •
  •
  •
end;
```

صحيح — لبنة — حقيقى

لاحظ الفاصلة والفاصلة المنقوطة

شكل (٨ ــ ٩)

أما عند الاستدعاء فتفصل جميع البارامترات الحقيقية بفاصلة عادية كالمثال الآتى :

```
demo(256, 73, '*', 'p', 'q', 'r', 26.45, 72.6, 18.253)
```

مثال (٨ ـــ ٣) البرنامج الابن والبرنامج الحفيد ا

فى هذا المثال يقرأ البرنامج عددين صحيحين موجبين ويطبعهما فى ترتيب تنازلى كما يضيف إلى العدد علامات الفاصلة لتمييز خانة الآلاف والملايين فالعدد **1235548** يكتب على الصورة **1,235,548** .

```
program compare(input,output);

var first, second : integer;

   procedure commawrite(n : integer);
   var millions, thousands, units : integer;

      procedure zerowrite(m : integer);
      begin
         if m < 100 then write('0');
         if m < 10 then write('0');
         write(m)
      end;
```

البرنامج الفرعي الحفيد

```
   begin
      millions := n div 1000000;
      thousands := (n mod 1000000) div 1000;
      units := n mod 1000;
      if millions > 0 then
      begin
         write(millions, ',');
         zerowrite(thousands); write(',');
         zerowrite(units)
      end
      else
         if thousands > 0 then
         begin
            write(thousands, ',');
            zerowrite(units)
         end
         else write(units);

      writeln
   end;
```

البرنامج الفرعي الابن

```
begin
   read(first, second);
   if first > second then
   begin
      commawrite(first);  commawrite(second)
   end
   else
   begin
      commawrite(second);  commawrite(first)
   end
end.
```

البرنامج الرئيسي

شكل (٨ ـــ ١٠)

ومن المفضل عند قراءة برنامج كبير أن نبدأ بقراءة البرنامج الرئيسى (وكذلك عند كتابة البرنامج) . فالبرنامج الرئيسى يبدأ بقراءة العددين **first, second** ثم يقارن بينهما باستخدام العبارة الشرطية لترتيبها تنازلياً . ثم يتولى البرنامج الفرعى commawrite طباعة الفواصل . عند إنشاء مثل هذا البرنامج نبدأ بهذه الخطوات المنطقية ثم نفكر بعد ذلك فى خطوات البرنامج الفرعى الذى سيطبع الفواصل .

وعند إنشاء البرنامج الفرعى commawrite سوف تتضح الحاجة إلى برنامج فرعى آخر بداخل البرنامج الفرعى الأول وهو البرنامج zerowrite وذلك لوجود أرقام صغيرة مثل 32,152 لا يتطلب طباعتها محتوية أصفاراً على يسارها .

والمفهوم الذى نخرج به من هذا المثال أن البرامج الفرعية قد تتداخل وتتعدد بحيث يكون هناك **برنامج رئيسى وبرنامج ابن وبرنامج حفيد** وهكذا .. وينطبق على البرنامج الابن والبرنامج الحفيد ما ينطبق على البرنامج الرئيسى والبرنامج الفرعى من حيث أن متغيرات البرنامج الابن تعتبر متغيرات عامة بالنسبة للبرنامج الحفيد ويمكن استخدامها بداخله .

وهذا لا يمنع أن تكون هناك عدة برامج فرعية غير مرتبطة ببعضها البعض وكلها تخدم برنامجاً رئيسياً واحداً كالمثال التالى :

مثال (٨ ــ ٤) ترجمة لغات الكومبيوتر:

نعلم أن لغات الكومبيوتر عالية المستوى ــ مثل باسكال ــ تحتاج إلى مترجم compiler لتحويل التعليمات المكتوبة بهذه اللغة إلى لغة الماكينة التى **تتسم بالتبسيط الشديد والوصف التفصيلى لكل خطوة من الخطوات المطلوبة من الكومبيوتر . فعلى سبيل المثال إذا اعتبرنا العملية الحسابية الآتية :**

$$a + x * z$$

فلكي نصف تنفيذ هذه العملية للكومبيوتر باستخدام لغة الماكينة فإننا نصفها بطريقة مماثلة للأسلوب الذى نشرح به خطوات استخدام الماكينة الحاسبة كالآتى :

١ ـــ أدخل قيمة المتغير x .
٢ ـــ اضغط على الزر (*) (زر الضرب) .
٣ ـــ أدخل قيمة المتغير z .
٤ ـــ اضغط على زر الجمع (+) .
٥ ـــ أدخل قيمة المتغير y .
٦ ـــ اطبع قيمة الناتج على الشاشة .

لاحظ أولوية عملية الضرب !

وللتسهيل سوف نفترض هذا العدد المحدود من المتغيرات والمؤثرات كما نفترض عدم استخدام الأقواس . ويتكون المتغير من حرف واحد بلا مسافات خالية بينه وبين أى لبنة أخرى .

وعلى البرنامج أن يحدّد أى المؤثرات يتعامل معها قبل الآخر (الأولوية) ، فالضرب والقسمة يأتيان قبل الجمع والطرح . وإذا تساوت أولوية مؤثرين فإن الأولوية تعطى لمن يأتى أولاً .

ويمكن التعبير عن هذا المنطق كالآتى :

إذا كان المؤثر الثانى أحد عناصر الفئة ['*','/']
والمؤثر الأول أحد عناصر الفئة ['+','-'] .
إذن فنفذ العملية المصاحبة للمؤثر الثانى قبل العملية المصاحبة للمؤثر الأول .
وإلاّ فنفذ العملية المصاحبة للمؤثر الأول قبل العملية المصاحبة للمؤثر الثانى .

وهذا المنطق يمكن وصفه بشريحة البرنامج التالية :

```
if (op2 in ['*', '/']) and (op1 in ['+', '-']) then      ← الشرط
begin
   valueof(var2);  }
   apply(op2);     }
   valueof(var3);  }  إجراء العملية الثانية
   apply(op1);     }
   valueof(var1)   }
end
else
begin
   valueof(var1);  }
   apply(op1);     }
   valueof(var2);  }  إجراء العملية الأولى
   apply(op2);     }
   valueof(var3)   }
end
```

شكل (٨ — ١١)

فى هذه الشريحة من البرنامج وصفنا المؤثر الأول بالمتغير **op1** والمؤثر الثانى بالمتغير **op2** . كما نلاحظ استخدام الفاصلة مع الفئة التى تحتوى على عناصر غير متصلة مثل ['+','-'] .

فإذا أتينا إلى تفصيلات تنفيذ العملية المصاحبة للمؤثر الثانى (العملية الثانية) نجد أن أول خطوة هى :

valueof(var2)

وبالطبع نتوقع أن يكون **valueof** هو اسم برنامج فرعى يقوم بإدخال قيمة المتغير **var2** . كما نلاحظ أن اسم المتغير قد استخدم كبارامتر تمهيداً لاستبداله ببارامتر حقيقى .

أما العبارة **apply** فهى تناظر برنامجاً فرعياً آخر يقوم بإصدار التعليمات

للضغط على زر معيّن من أزرار المؤثرات (/، -، +، *) والمؤثر المستخدم مع هذا البرنامج الفرعى هو البارامتر op2 . وهكذا ...

ويوضح شكل (٨ ـــ ١٢) البرنامج الكامل الذى يصف هذه العملية .

```
program evaluate(input, output);

var var1, op1, var2, op2, var3 : char;

  procedure valueof(variable : char);

  begin
     writeln('key in the value for ', variable)
  end;
```

أدخل قيمة المتغير

```
  procedure apply(op : char);

  begin
     write('press the ');
     case op of
        '+': write('addition');
        '-': write('subtraction');
        '*': write('multiplication');
        '/': write('division')
     end;
     writeln(' key')
  end;
```

اضغط على زر العملية المطلوبة

```
begin
   read(var1, op1, var2, op2, var3);

   if (op2 in ['*', '/']) and (op1 in ['+', '-']) the
   begin
      valueof(var2);
      apply(op2);
      valueof(var3);
      apply(op1);
      valueof(var1)
   end
   else
   begin
      valueof(var1);
      apply(op1);
      valueof(var2);
      apply(op2);
      valueof(var3)
   end;

   writeln('display the answer.')
end.
```

ترجمة لغات الكومبيوتر

شكل (٨ — ١٢)

(٨ — ٤) البارامتر المتغيّر variable parameter :

فى الفقرة السابقة كان البارامتر دائماً يصف **قيمة** معينة تتم معالجتها عندما يستدعى البرنامج الفرعى كالمثال الآتى :

```
procedure process(x : integer);
begin
    writeln(x);
    writeln(x*x)
end;
```

شكل (٨ — ١٣)

وعندما يستدعى هذا البرنامج الفرعى فإنه استبدال البارامتر x بأى تعبير صحيح مثل المتغيّر (a) أو قيمته العددية أو أى عنصر من عناصر مصفوفة عددية مثل mark[2,3] (كما فى شكل ٨ — ١٤) :

```
process(5742);
process(a);
process(a*5 + 32);
process(mark[2,3]);
process(mark[2,3]*3 + 564)
```

شكل (٨ — ١٤)

وعندما يستدعى البرنامج الفرعى بإحدى هذه العبارات فإن البارامتر الحقيقى يُرسل البيان المطلوب إلى البرنامج الفرعى ، وتتم المعالجة .

هذه النوعية من البارامترات يطلق عليها **البارامترات البسيطة (simple**

. parameters)

ولكنه فى بعض الأحيان قد يتطلب الأمر إرسال البيانات من البرنامج الفرعى إلى البرنامج الرئيسى ، وهذا يمكن تحقيقه بتزويد البرنامج الفرعى **ببارامتر متغيّر (variable parameter)** والذى يخبر البرنامج الفرعى : « إلى أين يرسل المعلومة المطلوبة بالتحديد » . ولنضرب مثلاً بالبرنامج الذى عرضناه فى التمرين (٤ — ١٠) عن جمع الأصوات فى الانتخابات (أنظر إجابة التمارين فى نهاية الكتاب) ، ولنستمر فى تطوير البرنامج حتى يطبع رسالة باسم الحزب الفائز .

والمنطق التالى يعبر عن البرنامج المطلوب :

١ — اجمع الأصوات لحزب واحد وخزّن الناتج فى المتغيّر **party1overall** .

٢ — اجمع الأصوات للحزب الثانى وخزّن الناتج فى المتغيّر **party2overall** .

٣ — **إذا** تساوى عدد الأصوات لكل من الحزبين اطبع النتيجة التالية «a draw» بمعنى تعادل !.

٤ — **وإلاّ** فإذا كان عدد أصوات الحزب الأول أكبر فاطبع النتيجة «a win for party 1.» أى فوز الحزب الأول .

٥ — وإلاّ فاطبع النتيجة «a win for party 2.» أى فوز الحزب الثانى .

ولو تأملنا الخطوتين (١) ، (٢) لوجدنا أن العملية المطلوبة بكل منهما واحدة وهى جمع الأصوات وتخزينها فى متغيّر والفارق الوحيد هو اسم المتغير الذى توضع فيه النتيجة .

فلننشيء إذن برنامجاً فرعياً وليكن اسمه **"addupvotesfor"** بمعنى « اجمع الأصوات للحزب ... » ولنتوقع أن نستخدمه بالطريقة الآتية :

```
addupvotesfor(party1overall);
addupvotesfor(party2overall)
```

فى الحالة الأولى يُستخدم لتجميع أصوات الحزب الأول .
وفى الحالة الثانية يُستخدم لتجميع أصوات الحزب الثانى .

أى أننا نطلب من البرنامج الفرعى أن يضع قيمة معينة فى المتغير المستخدم كبارامتر . وهذا يجب الإعلان عنه فى تعريف البرنامج الفرعى كما هو موضح بالمثال الآتى :

مثال (٨ ــ ٥) انتخابات ..

```
program election3(input,output);

var party1overall, party2overall : integer;

  procedure addupvotesfor(var total : integer);

  var next : integer;

  begin
     total := 0;
     read(next);
     repeat
        total := total + next;
        read(next)
     until next < 0
  end;
begin
  addupvotesfor(party1overall);
  addupvotesfor(party2overall);

  if party1overall = party2overall then
     writeln('a draw.')
  else if party1overall > party2overall then
          writeln('a win for party 1.')
       else writeln('a win for party 2.')
end.
```

البارامتر المتغير

البرنامج الفرعى لتجميع أصوات الأحزاب

شكل (٨ ــ ١٥)

وكما نلاحظ فى مستهل البرنامج الفرعى فإن كلمة **VAR** قد ظهرت فى تعريف البارامتر **total** .

ولذلك فعندما يُستدعى البرنامج الفرعى بالعبارة :

addupvotesfor(party1overall)

فإن المتغير **total** سوف يشير إلى نفس خانة الذاكرة التى يشير إليها المتغير **party1overall** .

ولذلك فإن تنفيذ العبارة الواردة فى البرنامج الفرعى :

total := total + next

يكافىء تماماً كما لو أن العبارة التالية هى التى نفذت :

party1overall := party1overall + next

أى أن الأصوات يتم تجميعها فى المتغيّر party1overall .

كذلك الحال بالنسبة لتجميع الأصوات فى المتغير الثانى party2overall .

ملاحظات :

● من الجدير بالذكر أنه بخلاف البارامترات البسيطة **لا يجوز** استدعاء البرنامج الفرعى باستخدام بارامتر ثابت مثل :

addupvotesfor(3);

أو باستخدام تعبير مثل :

addupvotesfor(party1overall + party2overall)

فالبرنامج الفرعى ينتظر منا اسم متغيّر (أو اسم متغيّر دليلى) حتى يضع فيه القيمة الناتجة من المعالجة .

فعلى سبيل المثال كان من الجائز إعلان مصفوفة مثل :

var party : array [1..3] of integer;

وفى هذه الحال يمكن استدعاء البرنامج الفرعى بعبارات كالعبارات الآتية :

```
addupvotesfor(party[1]);
addupvotesfor(party[2]);
addupvotesfor(party[3])
```

أو باستخدام حلقة تكرارية مثل :

```
for next := 1 to 3 do
    addupvotesfor(party[next])
```

● ومن الملاحظات التى يجب وضعها فى الاعتبار هى ضرورة فصل البارامترات المتغيّرة عن بقية البارامترات بفاصلة منقوطة . والمثال الآتى يعرض برنامجاً فرعياً يستقبل قيمتين **value 1** ، **value 2** ويضع مجموعهما فى متغير ثالث **sum** وحاصل ضربهما فى متغير رابع **product** شكل (٨ ــ ١٦) :

```
procedure addandmultiply(value1, value2 : real;
                         var sum, product : real);
begin
   sum := value1 + value2;
   product := value1 * value2
end;
```

لاحظ الفاصلة المنقوطة

شكل (٨ ــ ١٦)

ورغم أن البارامترات جميعاً حقيقية لكننا نلاحظ تقسيمها إلى بارامترات بسيطة ومتغيرة وفصل المجموعتين بفاصلة منقوطة .

وقد يستدعى مثل هذا البرنامج الفرعى بالطريقة الآتية بعد (شكل ٨ — ١٧) :

```
var x, y, sum1, prod1 : real;
    a, b : array [1..10] of real;
      .
      .
   addandmultiply(3.4, 7.2, x, y);
   addandmultiply(x + 3.7, y + 4.3, sum1, prod1);
   addandmultiply(23.72, 63.15, a[1], b[1])
      .
      .
```

شكل (٨ — ١٧)

(٨ — ٥) *البارامترات الدليلية* ***array parameters*** :

عرفنا من قبل المتغيرات الدليلية وهى المتغيّرات التى تمثل عناصر المصفوفة . وفى هذه الفقرة نعرض استخدام المتغيرات الدليلية كبارامترات للبرامج الفرعية .

وقبل أن نعرض تفصيلات الطريقة المستخدمة دعنا نرى أولاً الحاجة التى تدفعنا لاستخدام مثل هذه النوعية من البارامترات . ولنفرض أن لدينا درجات مجموعة من الطلبة تقدموا للامتحان فى ست مواد مختلفة ، والمطلوب كتابة برنامج يقوم بقراءة وحساب درجات المواد المختلفة والمجموع لكل طالب وطباعة هذه النتائج بحسب الترتيب : الأعلى فالأقل . ولنبدأ بطالبين هما **على** و**عمر** : وفى هذه الحالة يمكن توقع أن يعمل البرنامج كما فى شكل (٨ — ١٨) حيث يستقبل درجات كل من عمر وعلى ثم يطبع النتائج مرتبة .

تنفيذ البرنامج المتوقع :-

```
Running
8
7
6
7
8
5
4
7
8
8
7
9
ALY:     4  7  8  8  7  9   total: 43
OMAR:    8  7  6  7  8  5   total: 41

>
```

درجات الطالب "عمر"

درجات الطالب "علي"

المجموع الكلي .. بالترتيب

شكل (٨ ــ ١٨)

ولنبدأ بتصميم البرنامج الرئيسى .. وليكن هو البرنامج الموضح شكل (٨ ــ ١٩) .

```
var omarsmarks, alysmarks  :  array [1..6] of integer;
    omarstotal, alystotal  :  integer;
    .
    .
    .
begin
   readandadd(omarsmarks, omarstotal);
   readandadd(alysmarks, alystotal);
   if omarstotal > alystotal then
   begin
      write('OMAR: '); writeout(omarsmarks, omarstotal );
      write('ALY:  '); writeout(alysmarks,  alystotal  )
   end
   else
   begin
      write('ALY:  '); writeout(alysmarks,  alystotal );
      write('OMAR: '); writeout(omarsmarks, omarstotal);
   end
end.
```

شكل (٨ ــ ١٩)

وفى هذا البرنامج فإن وظيفة البرنامج الفرعى readandadd هى قراءة درجات المواد الستة وتخزينها فى مصفوفة الدرجات الخاصة بكل طالب ثم تجميع هذه الدرجات فى متغيّر مجموع الدرجات لكل طالب .

فمصفوفة درجات الطالب عمر هى :	omarsmarks
أما متغير مجموع الدرجات له فهو	omarstotal
ومصفوفة الدرجات للطالب على هى	alysmarks
ومتغير مجموع الدرجات هو	alystotal

ولذلك فإن البرنامج الفرعى سوف يتم تعريفه بواسطة هذين البارامترين : **مصفوفة من ستة عناصر ، ومتغير صحيح .**

ولكن فى مثل هذه الحالة لا يجوز استخدام التوصيف المعقّد **array[1..6]** لتوصيف بارامتر البرنامج الفرعى . بل يجب استخدام الأنماط المبتكرة بإضافة سطر جديد مثل :

type marklist = array [1..6] of integer ;

فى هذا الإعلان تم التعريف بالمصفوفة marklist التى تصلح لاستخدامها خلال البرنامج كمصفوفة عامة يمكن أن تحتوى درجات أى طالب .

وبذلك فإن البرنامج الفرعى لقراءة وتجميع الدرجات يصبح كالآتى :

```
procedure readandadd (var markfor  :  marklist ;
                     var total  :  integer);
var nextpaper  :  integer ;
begin
   total  := 0;
   for nextpaper  := 1 to 6 do
   begin
      readln (markfor[nextpaper]);
      total := total + markfor[nextpaper];
   end
end;
```

مصفوفة عامة

البرنامج الفرعى لقراءة وتجميع الدرجات

شكل (٨ — ٢٠)

كما أن المتغيّرات المعلنة فى البرنامج الرئيسى يمكن تبسيطها إلى الصورة التالية :

var omarsmarks, alysmarks : marklist ;
omarstotal, alystotal : integer ;

أى أن بارامتر المصفوفة أو البارامتر الدليلى يجب أن يكون دائماً فى صورة متغير **VAR** (إلاّ فى حالات قليلة تخرج عن حدود الكتاب) .

وهذا هو البرنامج الكامل باستخدام البارامترات الدليلية :

البرنامج الرئيسي

```
program exam (input, output);

type marklist = array [1..6] of integer ;

var omarsmarks, alysmarks  :  marklist ;
    omarstotal, alystotal  :  integer ;
```

البرنامج الفرعي لتجميع الدرجات

```
procedure readandadd (var markfor  :  marklist ;
                      var total   :  integer) ;
var nextpaper  :  integer ;
begin
   total  := 0;
   for nextpaper  := 1 to 6 do
   begin
      readln (markfor[nextpaper]);
      total := total + markfor[nextpaper];
   end
end;
```

البرنامج الفرعي للطباعة

```
procedure writeout (var markfor : marklist;
                    total : integer);
var nextpaper : integer;
begin
   for nextpaper := 1 to 6 do
      write(markfor[nextpaper]  :  3);
   writeln('   total: ', total)
end;
```

```
begin
   readandadd(omarsmarks, omarstotal);
   readandadd(alysmarks, alystotal);
   if omarstotal > alystotal then
   begin
      write('OMAR: '); writeout(omarsmarks, omarstotal
      write('ALY:  '); writeout(alysmarks,  alystotal
   end
   else
   begin
      write('ALY:  '); writeout(alysmarks,  alystotal )
      write('OMAR: '); writeout(omarsmarks, omarstotal)
   end
end.
```

شكل (٨ — ٢١)

(٨ ــ ٦) الدوال functions :

عرفنا من قبل الدوال القياسية الموجودة فى لغة باسكال مثل sqr ، sqrt . وفى هذه الفقرة سوف نعرض **الدوال المبتكرة** أى التى يقوم بإنشائها المبرمج وفقاً لاحتياجاته . والدالة تؤدى عملاً مشابهاً للبرنامج الفرعى ولكن لكل منهما استخداماً مناسباً . فلنر الفارق أولاً ما بين الدالة والبرنامج الفرعى .

تتشابه الدوال مع البرامج الفرعية فى أن كليهما يمثل جزءاً مستقلاً من وحدات بناء البرنامج (module) ويتم استدعاؤه باستخدام اسم ما . ولكن طريقة الاستدعاء تختلف لكل منهما . أما الفارق الأساسى بين النوعين فهو أن الدالة تستخدم لتوليد قيمة وحيدة تحل محل التعبير الذى استدعيت به الدالة .

ولنضرب مثلاً باستخدام الدالة القياسية sqrt فى التعبير الآتى :

y: = x + sqrt(2)

عند تنفيذ هذه العبارة فإن الدالة **sqrt** تقوم بتوليد قيمة عددية معينة تحل محل التعبير **sqrt(2)** فى العبارة .

فإذا كانت الدالة مبتكرة ، فسوف تؤدى عملها بنفس الطريقة فيما عدا أن المبرمج سوف يضيف إلى برنامجه برنامجاً صغيراً لتعريف الدالة المبتكرة ووصف العمل الذى تقوم به .

ولنضرب مثلاً ....

مثال (٨ ــ ٦) :

البرنامج الآتى يستقبل ثلاثة أزواج من الأرقام ويقوم بجمع الأرقام الكبرى فى كل زوج من الأزواج . شكل (٨ ــ ٢٢) .

```
program add(input,output);

var a,b,p,q,x,y : real;

   function max(first,second : real) : real;

   begin
      if first > second then max := first
      else max := second
   end
begin
   read(a,b, p,q, x,y);
   writeln(max(a,b) + max(p,q) + max(x,y))
end.
```

برنامج الدالة

بداية البرنامج الرئيسى

شكل (٨ ــ ٢٢)

وفى هذا البرنامج نرى برنامج الدالة قد جاء مباشرة بعد فقرة الإعلانات ــ تماماً كما البرنامج الفرعى ــ ووظيفة برنامج الدالة (أو الدالة) هنا هى مقارنة عددين first ، second وإيجاد أكبرهما ووضعه فى المتغير max الذى يمثل اسم الدالة .

ونلاحظ فى البرنامج الرئيسى أن استدعاء الدالة يتم باستخدام الاسم max الذى يعقبه بارامتران مثل :

max(x,y)

وعندما يتم تقييم هذا التعبير فإنه يستبدل بالكامل بالقيمة التى أنتجها برنامج الدالة عندما أجرى المقارنة بين العددين y و x .

أما فى تعريف الدالة نفسها ، فنرى أن الدالة قد تم تعريفها كما لو كانت متغيراً عادياً من النمط الحقيقى . كما أعلن عن نمط البارامترات المستخدمة معها (first,second) أيضاً .

وعند استدعاء الدالة فإن البارامتران first,second يُستبدلان بالبارامترات الحقيقية مثل a,b أو x,y,.... ومن الجائز استخدام بارامترات عددية مع الدالة مثل :

writeln(max(63.45, 61.23))

فى هذه الحالة فإن نتيجة العبارة تكون هى العدد 63.45 ومن الجائز أيضاً استخدام التعبيرات الحسابية أو الدوال كبارامترات (تماماً كما هو الحال مع الدوال القياسية) مثل :

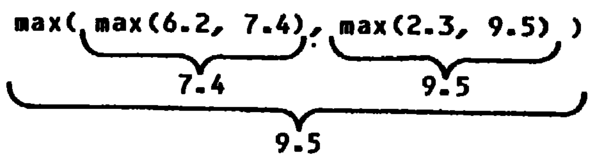
writeln(max(sqrt(2), sqrt(3))

وهذه العبارة سوف تعطى النتيجة 1.73 .

كما يجوز استخدام الدالة max نفسها كبارامتر وفى هذه الحالة فإنها تستدعى نفسها بنفسها كالمثال الآتى :

writeln(max(max(6.2, 7.4), max(2.3, 9.5))

هذه العبارة سوف يتم تقييمها كما فى شكل (٨ ــ ٢٣) لتنتج العدد 9.5 فى النهاية .



شكل (٨ ــ ٢٣)

● بقى أن نتذكر الترتيب المفروض للإعلانات التى يمكن آن يتضمنها أى برنامج ، حيث يجب أن يتبع هذا التسلسل :

١) إعلان الثوابت المسمّاه	CONST
٢) إعلان الأنماط المبتكرة	TYPE
٣) إعلان المتغيرات	VAR
٤) إعلان البرامج الفرعية والدوال المبتكرة .	Procedure Function



■ **تمارين على الباب الثامن :**

س (٨ ــ ١) اكتب برنامجاً لترتيب ثلاثة أعداد ترتيباً تنازلياً باستخدام المنطق الآتى :

١ ــ رتّب العدد الأول والعدد الثانى (وهذا قد يتضمن تبديلهما) .

٢ ــ رتّب العدد الأول والثالث (يحتوى العدد الأول الآن على أكبر قيمة) .

٣ ــ رتّب العدد الثانى والثالث .

استخدم فى عملية الترتيب برنامجاً فرعياً (وليكن اسمه order) بحيث يقوم بالمقارنات والتبديل ، ويستدعى من البرنامج الرئيسى كالآتى :

```
order(first,  second);
order(first,third);
order(second,  third)
```

س (٨ ــ ٢) اكتب برنامجاً لتقييم التعبير الآتى :

$$x^5 + y^4 + z^3$$

حيث تُستقبل قيم المتغيرات x,y,z كمدخلات للبرنامج . يجب أن يتضمن برنامجك الدالة (power) والتى يمكن استخدامها لرفع عدد حقيقى إلى أس صحيح . فعلى سبيل المثال :

ينتج عن استخدام الدالة power بالبارامترات الآتية بعد :

power(x,5)

القيمة x^5 .

س (٨ ــ ٣) اكتب برنامجاً لفرز الأعداد (sorting) الموضحة فى الجدول رقم

(1) لتحويلها إلى الجدول رقم (2) على أن يتضمن البرنامج البرامج الفرعية الآتية .

١ ـــ برنامج لطباعة الجدول رقم (1) .

٢ ـــ برنامج لفرز الأعداد بالجدول وترتيبها تصاعدياً (2) .

٣ ـــ برنامج لطباعة الجدول رقم (2) (يجوز أن يكون هو نفسه البرنامج الفرعى الأول) .

Array Contents:

1:	92	91	94	93	96	95	98	97	100	99
11:	82	81	84	83	86	85	88	87	90	89
21:	72	71	74	73	76	75	78	77	80	79
31:	62	61	64	63	66	65	68	67	70	69
41:	52	51	54	53	56	55	58	57	60	59
51:	42	41	44	43	46	45	48	47	50	49
61:	32	31	34	33	36	35	38	37	40	39
71:	22	21	24	23	26	25	28	27	30	29
81:	12	11	14	13	16	15	18	17	20	19
91:	2	1	4	3	6	5	8	7	10	9

الجدول 1

Array Contents:

1:	1	2	3	4	5	6	7	8	9	10
11:	11	12	13	14	15	16	17	18	19	20
21:	21	22	23	24	25	26	27	28	29	30
31:	31	32	33	34	35	36	37	38	39	40
41:	41	42	43	44	45	46	47	48	49	50
51:	51	52	53	54	55	56	57	58	59	60
61:	61	62	63	64	65	66	67	68	69	70
71:	71	72	73	74	75	76	77	78	79	80
81:	81	82	83	84	85	86	87	88	89	90
91:	91	92	93	94	95	96	97	98	99	100

الجدول 2

شكل (٨ ـــ ٢٤)

س (٨ ـــ ٤) اكتب برنامجاً لطبع الأعداد الأولية (primes) التى تنحصر بين العدد « صفر » والعدد "1000" .

ملاحظة الأعداد الأولية هى التى لا تقبل القسمة إلاّ على نفسها (انظر الشكل الذى يمثل نتيجة البرنامج) .

```
Running
          2
          3
         5
         7
        11
        13
        17
        19
        23
        29
        31
        37
        41
        43
        47
        53
        59
        61
        67
        71
        73
        79
         .
 >       .
         .
```

الأعداد الأولية

شكل (٨ ــ ٢٥)

ملاحــق الكتــاب

الملحق (أ) الكلمات المحجوزة
(Reserved Words)

```
and          function     program
array        goto         record
begin        if           repeat
case         in           set
const        label        then
div          mod          to
do           nil          type
downto       not          until
else         of           var
end          or           while
file         packed       with
for          procedure
```

ملاحظة : بعض الكلمات التى وردت هنا لم نتعرض لها فى هذا الكتاب .

الملحق (ب) الكلمات القياسية
Standard Words

Constants:	false, true, maxint
Types:	integer, real, boolean, char, text
Files:	input, output
Functions:	abs, arctan, chr, cos, eof, eoln, exp, ln, odd, ord, pred, round, sin, sqr, sqrt, succ, trunc
Procedures:	dispose, get, new, pack, page, put, read, readln, reset, rewrite, unpack, write, writeln
Directive:	forward

ويضيف طراز اللغة UCSD هذه المجموعة أيضاً :

Types:	string, interactive
Functions:	atan, (in place of arctan), log, pwroften, length, pos, concat, copy
Procedures:	delete, insert, str, close, seek, mark, release

ملاحظة : بعض الكلمات التى وردت هنا لم نتعرض لها فى الكتاب .

الملحق (جـ) الدوال القياسية فى لغة باسكال
PASCAL standard functions

الدالة	نمط الدليل (أو البارامتر)	نمط النتيجة	التعريف
abs	صحيح حقيقى	صحيح حقيقى	القيمة المطلقة
sqr	صحيح حقيقى	صحيح حقيقى	التربيع
sin	صحيح/حقيقى	حقيقى	الجيب (الطبيعى)
cos	صحيح/حقيقى	حقيقى	جيب التمام (الطبيعى)
exp	صحيح/حقيقى	حقيقى	الدالة الأسيّة e^x
Ln	صحيح/حقيقى	حقيقى	اللوغاريتم الطبيعى
sqrt	صحيح/حقيقى	حقيقى	الجذر التربيعى
arctan	صحيح/حقيقى	حقيقى	(ظا$^{-1}$) الزاوية التى ظلها
odd	صحيح	منطقى	أختبار الأعداد الفردية والزوجية
trunc	حقيقى	صحيح	دالة القطع
round	حقيقى	صحيح	دالة التقريب
ord	لبنة	صحيح	
chr	صحيح	لبنة	لتحويل العدد الترتيبى إلى اللبنة المناظرة
succ	لبنة	لبنة	لإيجاد اللبنة التالية فى الترتيب
pred فى الترتيب	لبنة	لبنة	لإيجاد اللبنة السابقة
eoln	ملف من اللبنات	منطقى	لاختبار نهاية السطر
eof	ملف	منطقى	لاختبار نهاية الملف

ملاحظة (١) :

- في حالة الجيب وجيب التمام يجب أن يكون الدليل بالتقدير الدائرى .
- في حالة « الزاوية التى ظلها » arctan تكون النتيجة بالتقدير الدائرى .

ملاحظة (٢) :

تدل كلمة صحيح/حقيقى أن دليل الدالة يمكن أن يكون صحيحاً أو حقيقياً أو تركيبة من الأعداد الصحيحة والحقيقية .



الملحق (د) المؤثرات فى لغة باسكال
Pascal operators

التعريف	نمط النتيجة	نمط المعامل	الأولوية	المؤثر
النفى المنطقى	منطقى	منطقى	1	**not**
الضرب	حقيقى/صحيح	حقيقى/صحيح	2	★
القسمة	حقيقى	حقيقى/صحيح	2	/
القسمة الصحيحة	صحيح	صحيح	2	**div**
باقى القسمة	صحيح	صحيح	2	**mod**
الضرب المنطقى	منطقى	منطقى	2	**and**
الجمع	حقيقى/صحيح	حقيقى/صحيح	3	+
الطرح	حقيقى/صحيح	حقيقى/صحيح	3	−
الجمع المنطقى	منطقى	منطقى	3	or
التساوى	منطقى	حقيقى/صحيح	4	=
عدم التساوى	منطقى	حقيقى/صحيح	4	<>
أقل من	منطقى	حقيقى/صحيح	4	<
أكبر من	منطقى	حقيقى/صحيح	4	>
أكبر من أو يساوى	منطقى	حقيقى/صحيح	4	>=
أقل من أو يساوى	منطقى	حقيقى/صحيح	4	<=
عضوية الفئة	فئة	صحيح/لبنة	4	**in**

ملاحظة :

- تدل كلمة صحيح/حقيقى أن المعاملات التى يؤثر عليها المؤثر يمكن أن تكون صحيحة أو حقيقية أو تركيبة من المعاملات الصحيحة والحقيقية .
- بعض المؤثرات المذكورة يمكن أن تؤثر على معاملات من أنماط أخرى ولكن ذلك لم يناقش فى الكتاب .



الملحق (هـ)
شرح المؤثرات المنطقية
(logical operators)

المؤثر المنطقى AND :

إذا أردت أن تشغل جهازاً كهربياً مثل التليفزيون فإنك لا بد أن تأخذ إجراءين هما :

١ ــ وضع القابس (الفيشة) فى منبع التيار الكهربى .
٢ ــ فتح الجهاز بواسطة مفتاح القفل والفتح ON/OFF .

فإذا أجريت أحد الخطوتين دون الأخرى لن تتحقق النتيجة المطلوبة وهى تشغيل جهاز التليفزيون .

أى أن شرطين لا بد أن يتحققا معاً لتحقيق النتيجة :

وهذه هى وظيفة المؤثر المنطقى AND فإذا رمزنا للشرط الأول بالمتغير A ، وإذا رمزنا للشرط الثانى بالمتغير B ، فإن التعبير الآتى يعبر عن عملية تشغيل جهاز التليفزيون (أو أى عملية مماثلة) :

A AND B

وقيمة هذا التعبير تصبح "TRUE" عندما تأخذ المتغيرات A ، B القيمة TRUE أيضاً . كما يمكن أن يشمل التعبير أكثر من متغيرين . ويمكن تلخيص ذلك بالجدول الآتى :

(لاحظ أن الحرف F هو اختصار كلمة false والحرف T هو اختصار كلمة true) .

A الشرط الأول	B الشرط الثانى	A AND B نتيجة العملية
F	F	F
F	T	F
T	F	F
T	T	T

ويسمى المؤثر AND مؤثر الضرب المنطقى .

المؤثر المنطقى OR :

بعض العمليات يمكن تحقيقها بأحد طريقتين مختلفتين . فمثلاً الحجرة التى لها بابان يمكن الخروج منها من أى من البابين . كذلك مصابيح الإضاءة فى بئر السلم يمكن تشغيلها بأى زر من الأزرار (المفاتيح) فى أى دور .

معنى ذلك **أن تحقق أى شرط من الشرطين (أو الشروط) يمكن أن يؤدى إلى النتيجة المطلوبة .**

فإذا كان زر النور فى أحد الأدوار هو A وفى دور آخر هو B فإن التعبير الآتى يعبر عن عملية إضاءة نور السلم :

A OR B

ويأخذ هذا التعبير القيمة TRUE عندما يكون أى من A أو B يحتوى على القيمة TRUE ويمكن تلخيص ذلك بالجدول الآتى :

A الشرط الأول	B الشرط الثانى	A OR B نتيجة العملية
F	F	F
F	T	T
T	F	T
T	T	T

ويسمى المؤثر OR مؤثر الجمع المنطقى .

(٥ ـ ٤ ـ ٣) المؤثر المنطقى NOT :

أما المؤثر المنطقى NOT فكما هو ظاهر من اسمه فإنه يقوم بعكس الشروط ، ولذلك فهو يسمى مؤثر النفى المنطقى .

فإذا كان المتغير A يحتوى على القيمة TRUE ، فإن NOT A تصبح قيمته FALSE والعكس بالعكس .

ويمكن التعبير عن ذلك بالجدول الآتى :

A	NOT A
T	F
F	T

الملحق (و)
حلول التمرينات المختارة

حل تمرينات الباب الأول

جـ (١ ـــ ١) ، جـ (١ ـــ ٢)

(a) جائز

(b) جائز

(c) جائز

(d) غير جائز

(e) جائز

(f) غير جائز (كلمة محجوزة)

(g) غير جائز (المسافة الخالية غير مسموح بها)

(h) جائز ولكن لا يوصى باستخدام كلمات قياسية

(i) جائز ولكن هذا يحرمنا من استخدام كلمة أخرى مثل datafile1 (أكثر من ثمانية حروف متشابهة)

(j) غير جائز لأن الاسم يجب أن يبدأ بحرف كما لا يجوز أن يحتوى على العلامة (-) .

حل تمرينات الباب الثانى

جـ (٢ ـــ ١) :

البرنامج التالى يحتوى على الإعلان المطلوب فى الفقرة الأولى من السؤال ونلاحظ به الآتى :

- درجات الامتحان نعبّر عنها بالمتغيّر العددى الصحيح mark .
- متوسط الدرجات نعبّر عنه بالمتغير الحقيقى anerage .
- النتيجة (ناجح/راسب) نعبّر عنها بالمتغير المنطقى pass .
- مستوى الامتحان نعبّر عنه بمتغير اللبنة level .

أما الفقرة الثانية من البرنامج فهى تحتوى على عبارات التخصيص المطلوبة وهى تبدأ بالعبارة BEGIN وتنتهى بالعبارة END .

```
PROGRAM example;
VAR mark    : integer;
    average : real;
    pass    : boolean;
    level   : char;
BEGIN
   mark := 51;
   average := 47.5;
   level := 'a'
   pass := true;
END.
```

شكل (١)

جـ (٢ ـــ ٢) :

التعبير	قيمة التعبير
16/5	3.2
16 div 5	3
16 mod 5	1
19/5	3.8
19 div 5	3
19 mod 5	4
8 div 3 * 3	6
7 + 5 div 3	8
13 - 5 mod 3	11

شكل (٢)

جـ (٢ — ٣) :

لاحظ استخدام المتغيرات الآتية فى الإعلان :

degtorad · ثابت مسمّى للتحويل من درجة إلى زاوية نصف قطرية .

elevation متغير حقيقى تمثيل زاوية الارتفاع Θ بالدرجات

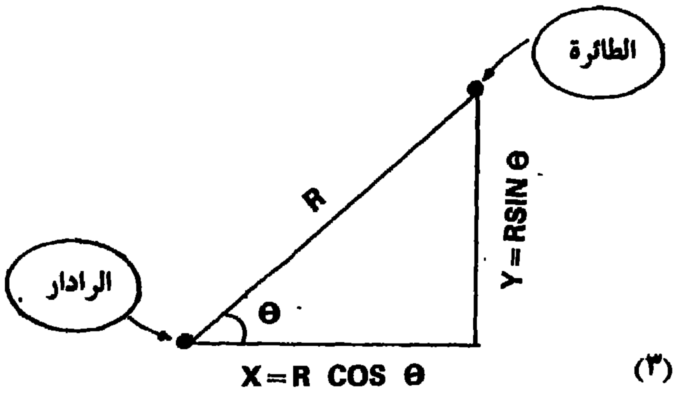
beamrage متغير حقيقى تمثيل المدى فى اتجاه الاشعاع الردارى R .

horizontal متغير حقيقى تمثيل البعد الأفقى الكارتيزى X .

angles متغير حقيقى تمثيل البعد الرأسى الكارتيزى Y .

radians متغير حقيقى تمثيل الزاوية Θ التقديرى النصف قطرى .

أنظر الرسم التالى لتسهيل المسألة من الناحية الفنية :



شكل (٣)

البرنامج

```
PROGRAM me29 (input, output);
CONST degtorad = 0.0174532925;
VAR elevation,
      beamrange,
      horizontal,
      angels,
      radians      : real;
BEGIN
   read (elevation, beamrange);
   radians := elevation * degtorad;
   horizontal := beamrange * cos (radians);
   angels := beamrange * sin (radians);
   writeln ('bandits at ', horizontal , ' units ');
   writeln ('angels ', angels );
END.
```

شكل (٤)

جـ (٢ ــ ٤) :

```
program journey(input,output);

var dist, mph, startime, hours, mins : integer;
    realhours : real;

begin

   read(dist, mph, startime);

   hours     := startime div 100;
   mins      := startime mod 100;
   realhours:= hours + mins/60;
```

حساب زمن الوصول كعدد حقيقي
من الساعات بعد منتصف الليل

```
   realhours:= realhours + dist/mph;
```

تحويل الزمن إلى ساعات ودقائق

```
   hours     := trunc(realhours);
   mins      := round((realhours - hours)*60);
   write(' arrival time: ', hours, '-', mins)

end.
```

شكل (٥)

حل تمرينات الباب الثالث

جـ (٣ — ١) :

```
program electricitybill2(input,output);

const unitrate = 3.4;
      standcharge = 1.14;
      asterisks = '******************************';

var present, previous : integer;

begin

   read(present,previous);
   writeln(asterisks);
   writeln('present meter reading ', present:7);
   writeln('previous meter reading', previous:7);
   writeln('units used            ',
                  present - previous :7);
   writeln('rate per unit         ',
                  unitrate :6:1, 'p');
   writeln('standing charge      £', standcharge:7:2);
   writeln('sum due is           £',
    (present-previous)*unitrate/100+standcharge:7:2);
   writeln(asterisks)

end.
```

شكل (٦)

جـ (٣ — ٢)

البرنامج

```
var a,b,c,d:integer:
var av:real;
begin
  read (a,b,c,d);
  writeln;
  av:=(a+b+c+d)/4;
  writeln('marks are: ',a,',',b,',',c,',',d):
  writeln('average is: ',av)
end.
```

التنفيذ

الأرقام المدخلة

```
10 20 30 40
marks are: 10,20,30,40
average is:   2.5000000000E+01

>
```

النتائج

شكل (٧)

جـ (٣ ـ ٣)

البرنامج

```
program sumandproduct:
var a,b:integer;
begin
  read (a,b);
  writeln;
  writeln(a,'+',b,'=',a+b);
  writeln(a,'*',b,'=',a*b)
end.
```

التنفيذ

```
5 6      ← الأرقام المدخلة
5+6=11   ← النتائج
5*6=30

>
```

شكل (٨)

```
~am payroll (input,output);
hours,payrate,overtime:integer;
gross:real;

n
read(hours,payrate,overtime);
writeln;
gross:=hours*payrate+overtime*payrate*1.5;
writeln('gross pay is:','$',gross)
```

البرنامج

التنفيذ

```
10 10   ← المدخلات
ss pay is:$  5.5000000000E+02
```

إجمالى الأجر

شكل (٩)

جـ (٣ — ٨)

بعد إعلان المتغيرات يتم فى هذا البرنامج قراءة طول الجسم ساكناً ، و بالعبارة read .

ونلاحظ أسماء المتغيرات الآتية المستخدمة فى البرنامج :

lnew طول الجسم أثناء الحركة .

lrest طول الجسم ساكناً .

v سرعة الجسم

c سرعة الضوء (ثابت مسمى)

```
PROGRAM fitzgerald (input, output);
CONST c = 2.99792458e+8;
VAR  lrest,
     lnew,
     v      : real;
BEGIN
     read (lrest);
     read (v);
     lnew := lrest * sqrt
                    (1.0 - sqr (v) / sqr (c));
     writeln (lnew)
END.
```

شكل (١٠)



حل تمرينات الباب الرابع

جـ (٤ ـــ ١) :

البرنامج

```
program multiplication(input, output);
var x,i:integer;
begin

read(x);
writeln;
for i:=1 to 10 do
  writeln(i,'x',x,'=',i*x)

end.
```

التنفيذ

العدد المُدخل

```
5
1x5=5
2x5=10
3x5=15
4x5=20
5x5=25
6x5=30
7x5=35
8x5=40
9x5=45
10x5=50

>
```

جدول الضرب للعدد 5

شكل (١١)

جـ (٤ ـــ ٢) :

البرنامج

```
program seven (input,output);

var n,i:integer;
begin
  read (n);
  writeln;

  for i:=1 to n do
  write('*');

  writeln;

  for i:=1 to n-1 do
  writeln('*':n-i)
end.
```

شكل (١٢)

هذا البرنامج يقوم برسم الرقم 7 بالنجوم التى يتوقف عددها (وبالتالى حجم الرقم 7 على المتغير n .

ويبدأ البرنامج بقراءة العدد n يليه طباعة سطر خال ثم تبدأ حلقة تكرارية لرسم السطر الأول من النجوم وقد استخدمنا فيها العبارة write التى تتميز بعدم الانتقال إلى السطر التالى بعد إتمام الطباعة .

وبعد انتهاء هذه الحلقة يتم الانتقال إلى السطر التالى بواسطة العبارة writeln يليها الحلقة التكرارية التى ترسم الخط المائل الممثل لجسم الرقم 7 ونلاحظ أن الصيغة المستخدمة للطباعة هى (n−i) وهى تجعل المسافة التى تظهر عندها النجمة * تنكمش عند كل سطر تال وهذا هو التنفيذ لقيمتين مختلفتين للمتغير n :

تنفيذ البرنامج

قيمة n المدخلة

```
8
********
      *
     *
    *
   *
  *
 *
*

>
```

```
16
****************
              *
             *
            *
           *
          *
         *
        *
       *
      *
     *
    *
   *
  *
 *
*

>
```

شكل (١٣)

جـ (٤ — ٣) :

البرنامج

```
program triangle (input,output);

var n,i:integer;
begin
  read (n);
  writeln;
  writeln('*':n);

  for i:=1 to n-2 do
  writeln('*':n-i,'*':i*2);

  for i:=1 to 2*n-1 do
  write('*')
end.
```

التنفيذ لقيم مختلفة للمتغير n

```
8
       *
      * *
     *   *
    *     *
   *       *
  *         *
 *           *
***************
>
```

قيمة n المدخلة

```
16
               *
              * *
             *   *
            *     *
           *       *
          *         *
         *           *
        *             *
       *               *
      *                 *
     *                   *
    *                     *
   *                       *
  *                         *
 *                           *
*******************************
>
```

شكل (١٤)

جـ (٤ ـ ٦) :

```
program month(input,output);
var m:integer;
begin
 writeln('please enter month number');
 read (m);
 writeln;
 case m of
   1:write(' january');
   2:write(' february');
   3:write(' march');
   4:write(' april');
   5:write(' may');
   6:write(' june');
   7:write(' july');
   8:write(' august');
   9:write(' september');
   10:write(' october');
   11:write(' november');
   12:write(' december');
 end;
write('.. this month contains: ');
case m of
   1,3,5,7,8,10,12:writeln('31 days');
   4,6,9,11:writeln('30 days');
   2:writeln('28 or 29 days depending on leap
   years');
end
end.
```

شكل (١٥)

تنفيذ البرنامج :

```
please enter course number
10
 course 10  fri  9am   fri 11am
>
```

ج (٤ — ٨) :

البرنامج :

```
program month(input,output);
var course:integer;
begin

    writeln('please enter course number');
    read (course);
    writeln;

    case course of
      1,2:write(' course ',course,'  thu   9am  fri 10am');
      3:  write(' course ',course,'  mon  10am  thu 10am');
      4,5:write(' course ',course,'  mon  11am  tue 11am');
      6,7:write(' course ',course.'  tue   9am  wed  2pm');
      8:  write(' course ',course,'  mon  12am  thu  9am');
      9:  write(' course ',course,'  tue  10am  wed 11am');
      10: write(' course ',course,'  fri  9am   fri 11am');
    end;

   end.
```

شكل (١٦)

ج (٤ — ١٠) :

```
program election2(input,output);

var party1next, party1overall,
    party2next, party2overall :integer;

begin
   party1overall := 0;
   read(party1next);
   repeat
      party1overall := party1overall + party1next;
      read(party1next)
   until party1next < 0;

   party2overall := 0;
   read(party2next);
   repeat
      party2overall := party2overall + party2next;
      read(party2next)
   until party2next < 0;

   writeln('party1: ', party1overall, '
           'party2: ', party2overall)
end.
```

شكل (١٧)

حل تمرينات الباب الخامس

```
program salesanalysis(input,output);

var dept, largesales : integer;
    nextitemprice : real;

begin

   for dept := 1 to 63 do
   begin
      largesales := 0;
      read(nextitemprice);
      repeat

         if nextitemprice >= 10 then
            largesales := largesales + 1;

         read(nextitemprice);
      until nextitemprice < 0;
      writeln('dept no. ', dept, ' has made '
            largesales, ' large sales')
   end

end.
```



شكل (١٨)

فى هذا البرنامج استخدمت المتغيرات الآتية :

dept متغير العدّاد لأقسام الشركة .

largesales متغير لتجميع السلع التى يزيد سعرها عن 10 جنيهات .

nextprice متغير حقيقى لتمثيل سعر السلع المختلفة .

ويتم التمييز بين كل قسم وآخر برقم القسم الموجود فى متغيّر العدّاد dept .

جـ (٥ — ٢) :

```
program tolerance4(input,output);

const standard = 6.37;
var length : real;
    noabove, nobelow, nowithin : integer;

begin
   nowithin := 0;  noabove := 0;
   nobelow := 0;   read(length);
   repeat
      if abs(length - standard) < 0.1 then
         nowithin := nowithin + 1
      else
         if length - standard >= 0.1 then
            noabove := noabove + 1
         else
            nobelow := nobelow + 1;
      read(length)
   until length < 0;
   writeln(nowithin, ' values are within tolerance.');
   writeln(noabove,  ' values are too large.');
   writeln(nobelow,  ' values are too small.')
end.
```

فرز الأطوال غير المقبولة

شكل (١٩)

المتغيرات المستخدمة فى البرنامج هى :

(١) ثوابت : standard تمثيل الطول القياسى 6.37 .

(٢) متغيرات حقيقية : length تمثيل الطول .

(٣) متغيرات صحيحة :

noabove عدد الأطوال الأكبر من اللازم .

nobelow عدد الأطوال الأصغر من اللازم .

nowithin عدد الأطوال المقبولة .



حل تمرينات الباب السادس

جـ (٦ — ١) :

```
program codeshift(input,output);

const shift = 5;

var character : char;  ordinal : integer;

begin

   repeat

      while not eoln(input) do
      begin
         read(character);
         if character in ['a'..'z'] then
         begin
            ordinal := ord(character);
            ordinal := ordinal + shift;
            if ordinal > ord('z') then
               ordinal := ordinal - 26;
            write(chr(ordinal))
         end
         else write(character)
      end;

      readln;  writeln
   until eof(input)

end.
```

شكل (٢٠)

جـ (٦ — ٣) :

```
program countwords(input, output);

var nextch : char;
    noofletters, noofwords, noof4words : integer;

begin
   noofwords := 0; noof4words := 0;

   repeat

      repeat read(nextch) until nextch in ['a'..'z'];


      noofletters := 0;
      repeat
         noofletters := noofletters + 1;
         read(nextch)
      until not (nextch in ['a'..'z']);

      noofwords := noofwords + 1;
      if noofletters = 4 then noof4words := noof4words
   until nextch ='.';

   writeln('words in sentence: ', noofwords);
   writeln('4-letter words in sentence: ', noof4words)
end.
```

شكل (٢١)

حل تمرينات الباب السابع

جـ (٧ — ١) :

```
program chararrays(input,output):
VAR ch: array[1..20] of char;
     n,i:integer;

BEGIN                      قراءة عدد الحروف
     readln(n);  ←
     for i:=1 to n do  ← قراءة الحروف
     read(ch[i]);
                           سطر خال
     writeln;  ←
     for i:=1 to n do  ← طباعة الحروف
     writeln(ch[i]);
end.
```

شكل (٢٢)

جـ (٧ — ٢) :

```
program salesanalysis4(input,output);

var salesfor : array [1..8] of integer;
   dept, col, scaledtotal : integer;

begin
   for dept := 1 to 8 do
      salesfor[dept] := 0;

   read(dept);
   repeat
      salesfor[dept] := salesfor[dept] + 1;
      read(dept);
   until dept = -1;

   for dept := 1 to 8 do
   begin
      scaledtotal := salesfor[dept] div 10;
      for col := 1 to scaledtotal do
         write('*');
      writeln
   end;
end.
```

شكل (٢٣)

جـ (٧ — ٣) :

شكل (٢٤)

```
program popmap2(input,output);

var popmap : array [1..8,1..10] of integer;
    row, col, rowa, cola, rowb, colb, subtotal:integer;

begin
   for row := 1 to 8 do
      for col := 1 to 10 do
         read(popmap[row,col]);

   subtotal := 0;
   read(rowa, rowb, cola, colb);

   for row := rowa to rowb do
      for col := cola to colb do
         subtotal := subtotal + popmap[row,col];

   writeln('population of sub-zone is ', subtotal)
end.
```

تجميع عدد سكان المنطقة الجزئية

جـ (٧ ـــ ٦)

```
program times(input,output);

var day, period, room, lecture : integer;
    place : array [1..5,1..6] of integer;

begin
```

شحن المتغيرات بقيمة ابتدائية (صفر)

```
   for day := 1 to 5 do
      for period := 1 to 6 do
         place[day,period] := 0;
```

إدخال البيانات

```
   for lecture := 1 to 20 do
   begin
      read(day,period,room);
      place[day,period] := room;
   end;
```

الطباعة

```
   writeln('period':28);
   writeln('1  2  3  4  5  6':34);
   writeln('----------------':34);

   for day := 1 to 5 do
   begin
      case day of
         1 : write('mon');
         2 : write('tue');
         3 : write('wed');
         4 : write('thu');
         5 : write('fri')
      end;

      write('            ');
      for period := 1 to 6 do
         if place[day,period] = 0 then
                  write('   ')
         else  write(place[day,period]:3);

      writeln
   end
end.
```

جدول المحاضرات

شكل (٢٥)

جـ (٧ ـــ ٨)

```
program terms:
    type term=(first.second.summer):
    var  thisterm,nextterm: term;

begin
  {نفترض أنه قيمة ما قد أعطيت للمتغير thisterm}
    if thisterm=summer
    then
       nextterm:=first
    else
       nextterm:=succ(thisterm);

end.
```

شكل (٢٦)

جـ (٧ ـــ ٩) :

```
        PROGRAM matrices (input, output);
        CONST   rowlim = 3;
                collim = 5;
TYPE     matrix = ARRAY [1..rowlim] OF
                        ARRAY[1..collim] OF real;
(* or   ARRAY [1..rowlim, 1..collim ] OF real; *)
VAR      a, b, c : matrix;
         row     : 1..rowlim;
         col     : 1..collim;
BEGIN
   (* assuming values given to a and b *)
   FOR row:= 1 TO rowlim DO
      FOR col:=1 TO collim DO
         c[row,col] := a[row,col] + b[row,col]
END.
```

شكل (٢٧)

جـ (٧ ـ ١٠) :

```
PROGRAM frequencies (input, output);
TYPE  alfachar = 'a'..'z';
VAR   ch : char;
      index: alfachar;
      table: ARRAY [alfachar] OF integer;
BEGIN
   FOR index := 'a' TO 'z' DO table[index]:= 0;
   WHILE NOT eof DO
   BEGIN
      read(ch);

         IF (ch >= 'a') AND (ch <= 'z')
            THEN table[ch]:= table[ch] + 1
      END;
      FOR index:='a' TO 'z' DO
         writeln ( index,' occurs ',
                              table[index],'
         times ')

  ..END.
```

شكل (٢٨)



إجابات تمارين الباب الثامن

ج (٨ — ١) :

```
program sort3numbers(input,output);

var first, second, third : integer;

  procedure order(var a,b : integer);

  var temp : integer;

  begin                                    البرنامج الفرعي
     if a < b then
     begin
        temp := a;
        a := b;
        b := temp
     end
  end;
begin
   read(first, second, third);
   order(first, second);                  البرنامج الرئيسي
   order(first, third);
   order(second, third);
   writeln(first:6, second:6, third:6)
end.
```

شكل (٢٩)

جـ (٨ ــ ٣) : فرز مصفوفة أعداد

```
VAR
  i, j, k: INTEGER;
  tab: ARRAY [1..100] OF INTEGER;

PROCEDURE printarr;                    ←  طباعة الجداول
{procedure to write out the array}

BEGIN
  WRITELN;  WRITELN (' Array Contents:');
  FOR i := 0 to 9 DO
  BEGIN
    WRITE (10*i+1:5, ':');
    FOR j := 1 TO 10 DO WRITE (tab[10*i+j]:5);
    WRITELN;
  END;
END;

PROCEDURE sortarr;                 ←  فرز الأعداد
{ Procedure to sort the array }

VAR
  temp: INTEGER;  swap: BOOLEAN;

BEGIN
  REPEAT
    swap := FALSE;
    FOR i := 1 to 99 DO
      IF tab[i] > tab[i+1] THEN
      BEGIN
        temp := tab[i];
        tab[i] := tab[i+1];
        tab[i+1] := temp;
        swap := TRUE;
      END;
  UNTIL NOT swap;
END;

BEGIN
  { Initialize the table that will be sorted }
  k := 0;
  FOR i := 9 DOWNTO 0 DO
    FOR j := i*10+1 TO (i+1)*10 DO
    BEGIN
      k := SUCC(k);
      IF ODD(j) THEN tab[k] := j+1 ELSE tab[k] := j-1;
    END;

  printarr; طباعة الجدول 1
  sortarr; الفرز
  printarr; طباعة الجدول 2
END.
```

شكل (٣٠)

ج (٨ — ٤)

توليد وطبع الأعداد
الأولية حتى 1000

```
var
  prime:       integer;
  rprime:      real;
  i:           integer;
  sqrtp:       integer;
  notprime:    boolean;

begin
  writeln('      2');
  writeln('      3');
  prime := 5;
  repeat
    rprime := prime;
    sqrtp := trunc(sqrt(rprime) + 1.0);
    i := 1;
    notprime := false;
    while (i < sqrtp) and (not notprime) do
      begin
        i := i + 2;
        notprime := (prime mod i = 0);
      end;
    if (not notprime) then writeln(prime:6);
    prime := prime + 2;
  until (prime > 10000);
end.
```

شكل (٣١)

وإلى اللقاء دائماً ...

رغم جولتنا الطويلة بين عبارات ودوال وتعبيرات اللغة ، فمازال فى لغة باسكال الكثير ، لاسيما أن ما جدّ عليها من تطوير وإضافات جعلها أضخم من أن يضمّها كتاب واحد .

لكننا حاولنا فى هذا اللقاء مع قارئنا العزيز أن نقدم له نخبة نافعة من أدوات ووسائل اللغة تمكنّه من بناء برنامج جيد متكامل الإنشاء وتغنيه عن كتاب آخر فى وقت قريب . أما الأجزاء التى أغفلناها فهى إما إضافات غير أساسية مثل عبارات الموسيقى وبعض عبارات الرسم ، وإما موضوعات اختلفت فيها الطرازات وأصبح من الضرورى الرجوع لكتاب اللغة الخاص بالكومبيوتر المعيّن ، وهذا ينطبق على موضوع الملفّات .

أما موضوع كتابنا القادم عن لغة باسكال بإذن الله ، فسوف يخدم فنون البرمجة أكثر مما يخدم مفردات اللغة وقواعدها . فالمهارة فى استخدام أدوات اللغة تجعل المبرمج يصل إلى الهدف المنشود من أقصر الطرق .

أما اللقاء القادم فلعله يكون حول أحد موضوعات الساعة فى مجال الكومبيوتر لا سيما وأننا على وشك أن ندخل عصراً جديداً من عصور الكومبيوتر وهو عصر الكومبيوتر ذى الخلايا العصبيّة !

وإلى اللقاء دائماً ...

أسامة الحسينى

المراجـــع

١ — **Computer Pragramming in Pascal**
DAVID LIGHTFOOT (Teach yourself)

٢ — **Simple Pascal**
JAMES J MCGREGOR,
ALAN H WATT (PITMAN)

٣ — **Proverbs for Programming in Pascal**
LOUISE E. MOSER,
ANDREW A. TURNBULL (Wiley)

كتب للمؤلف فى مجال الكومبيوتر

صدرت عن مكتبة ابن سينا

١ ـ *تحدث مع الكومبيوتر بلغة كوبول*

... المستوى الأول

٢ ـ *كل شيء عن الكومبيوتر (وكتابة البرامج بلغة بيسك)*

... مبسط للنشء ولأولياء الأمور .

٣ ـ *تحدث إلى الكومبيوتر بلغة بيسك*

... حتى المستوى المتقدم من لغة بيسك يضم اللغة القياسية قديمها وحديثها وأيضاً أشهر طرازات لغة بيسك .

٤ ـ *كيف يفكر الكومبيوتر*

... خرائط التسلسل المنطقى للبرامج والنظم الآلية وتحويل النظم اليدوية إلى آلية .

٥ ـ *برمجة الألعاب الكومبيوترية*

... طرق برمجة القذائف والتصادم والمؤثرات الصوتية مشروحة بلغة الطرازات الشهيرة للكومبيوتر المنزلى فى مصر والعالم العربى علاوة على لغة بيسك القياسية (ميكروسوفت) .

٦ ـ *مدخلك إلى عالم الكومبيوتر (المقدمة الأساسية لعلوم الكومبيوتر)*

٧ ـ *تعلم لغة الكومبيوتر سى من خلال لغة بيسك*

... مدخل مناسب للهواة والمحترفين لإجادة لغة سى .

٨ ــ الرسم بالكومبيوتر

... يتناول كل ما يخص استخدام الكومبيوتر فى الرسم .. يشرح عبارات بيسك القياسية للرسم الدقيق علاوة على أهم اللهجات المنتشرة للأجهزة الكومبيوتر المنزلى .

٩ ــ تحدث إلى الكومبيوتر بلغة لوجو

... لغة أصدقاء الروبوط ..

مع تطبيقات مختلفة فى البرمجة والألعاب باستخدام السلحفاه الشهيرة للرسم على الشاشة (turtle graphics) المدخل المناسب للأطفال إلى عالم الكومبيوتر .

١٠ ــ تحدث إلى الكومبيوتر بلغة فورتران ٧٧

... مرجعك العربى فى لغة فورتران يبدأ من البدايات الأولى للغة ويصل حتى مستويات متقدمة فى إنشاء البرامج . يضم الكتاب كل عبارات اللغة قديمها وحديثها مع تطبيقات على مختلف أجهزة الكومبيوتر .

١١ ــ برامج وألعاب كومبيوترية مشروحة (بلغة بيسك)

... برامج تعليمية فوازير ألعاب حروب وقذائف ومغامرات .. علاوة على برامج الملفات ومعالجة الكلمات بلغة بيسك . على أشهر طرازات الكومبيوتر المنزلى : تكساس ، كومودور ، أتارى ، BBC ، إليكترون ، سنكلير ..

١٢ ـــ قبل أن تشترى كومبيوتر

... دليلك فى شراء جهاز كومبيوتر لمنزلك أو مكتبك أو محلّك التجارى . دليلك فى التدريب إذا أردت العمل فى أحد مجالات الكومبيوتر نقد وتحليل خصائص أجهزة الكومبيوتر الشخصية والمنزلية .

وفى مجال الهندسة الكهربية :

ـــ كل شىء عن الإليكترونيات .

وفى مجال قصص الخيال العلمى للشباب :

ـــ إعدام إنسان آلى .

ـــ الدخول فى الثقب الأسود .

ـــ الدخول فى الثقب اللاأسود .

ـــ المعلوم والمجهول .



منشورات للمؤلف صدرت فى الولايات المتحدة الأمريكية بالاشتراك مع آخرين

(۱) تصميم وسيلة كومبيوترية لمعاونة المعوقين تعمل بالتعرف على الصوت البشرى باستخدام دوائر الخلايا العصبية .

بالاشتراك مع :

الأستاذة الدكتورة / جين ميردوك (جامعة نيو أورليانز)

الأستاذ الدكتور / عبد الفتاح الحسينى

(كبير مهندسى شركة تكنولوجى انترناشيونال)

المهندس / رودريجو رودريجس

(مهندس بشركة تكنولوجى انترناشيونال)

الناشر : IEEE

المؤتمر السنوى لرابطة مهندسى الكهرباء والإليكترونيات ـــ كاليفورنيا ١٩٨٨ .

J.Y. Murdock, A.A. Husseiny, E. Liang, O.A. Husseiny, R.J. Rodriguez, Improvement on Speech Recognition and Synthesis for Disabled Individuals Using Fuzzy Neural Net Retrofits, IEEE Annual International Conference on Neural Networks, San Diego, CA (1988).

(٢) محادثة الكومبيوتر بالصوف فى أغراض التعليم .

بالاشتراك مع :

الأستاذة الدكتورة / جين ميردوك

الدكتور / إنجو ليانج (مهندس بشركة تكنولوجى انترناشيونال)

الأستاذ الدكتور / عبد الفتاح الحسينى

الناشر :

المؤتمر السنوى الأول لجمعية دوائر الخلايا العصبية

(Neural Network Society)

بوسطن ، ماساتشوستس ١٩٨٨

J.Y. Murdock, O.A. Husseiny, E. Liang, A.A. Husseiny, A High Confidence Voice Interactive Hybrid Neural System for Learning, First Annual Meeting, International Network Society, Boston, MA (1988)

(٣) التحكم فى الربوط للعمل فى المناطق الخطرة

O.A. Husseiny, R.J. Rodriguez, E. Liang, A.A. Husseiny, Programmable Controller for Robotic Systems in Hostile Environment, Nuclear Technology, (1988).

بالاشتراك مع :

١ ـ المهندس / رودريجو رودريجس
٢ ـ الدكتور / إنجو ليانج
٣ ـ الأستاذ الدكتور / عبد الفتاح الحسينى

الناشر :

مجلة الصناعة النووية ١٩٨٨ Nuclear Technology

(٤) التحكم فى الربوطيات الصناعية

بالاشتراك مع :

١ ـ الدكتور / إنجو ليانج
٢ ـ الأستاذ الدكتور / عبد الفتاح الحسينى
٣ ـ المهندس / رودريجو رودريجس

الناشر :

مجلة « الصناعة باستخدام الروبوطيات والكومبيوتر » ١٩٨٨

Robotics & computer - Integrated Manufaturing

E. Liang, A.A. Husseiny, R.J. Rodriguez, O.A. Husseiny, Supervisory Controller for Robotics in Manufacturing, Robotics and Computer-Integrated Manufacturing, (1988).

(٥) تصميم وسيلة تعليمية للمعوّقين تعمل بالتعرف على الصوت البشرى .

بالاشتراك مع :

١ ــ المهندس / ميكل بارنت

٢ ــ الأستاذة الدكتورة / جين ميردوك

٣ ــ الأستاذ الدكتور / عبد الفتاح الحسينى

الناشر :

تقرير وزارة التعليم بالولايات المتحدة ١٩٨٨ .

O.A. Husseiny, Michael Barnett, contributions to Jane Y. Murdock, A.A. Husseiny, A User-Friendly Voice Interactive Learning Aid for Individuals with Handicaps (VILAH), TII Rep. # TILA/7088100/R and DOEd/870088-6 (1988).

(٦) *التحكم بالصوت والصورة فى معامل اختبار الروبوطيات*

بالاشتراك مع :

١ ــ المهندس / ميكل بارنت

٢ ــ المهندس / ريتشارد جاركا

٣ ــ الأستاذة الدكتورة / زينب صبرى

٤ ــ المهندس / راندى مانيجولت

٥ ــ المهندس / جون بوش

الناشر :

تقرير شركة تكنولوجى انترناشيونال ١٩٨٨ .

O.A. Husseiny, contributions to Richard E. Jarka, Zeinab A. Sabri, S. Keith Adams, A.A. Husseiny, Randy Manigault, John Bush, Development of Requirements for an Advanced Robotic Laboratory Equipment, TII RD&E Center Technical Report # 12680 (1988).

(٧) تلافى الأخطاء عن طريق جهاز التحكم فى الربوط

بالاشتراك مع :

١ ـــ الأستاذ الدكتور / عبد الفتاح الحسينى

٢ ـــ الدكتور / إنجو ليانج

٣ ـــ المهندس / رودريجو رودريجس

الناشر :

مجلة « روبوطيكا » ١٩٨٨

A.A. Husseiny, E. Liang, R.J. Rodriguez, O.A. Husseiny, Fault-Tolerant Robotic Controller, Robotica, (1988).

فهرست الكتاب

■ الباب الثالث : إدخال البيانات ـــ طباعة المعلومات
(Input-Output)

■ الباب الرابع : التفريع والتكرار *(Branching & Looping)*

■ الباب السابع : المصفوفات (arrays)

■ الباب الثامن : البرامج الفرعية والدوال
procedures & functions

■ ملاحق الكتاب

رقم الإيداع ٧٦٠٠/ ٨٨

دار النصر للطباعة الإسلامية
٢ ـ شارع نشاطى شبرا القاهرة
ت: ٧٧٣٢٢١

كلمة غلاف المؤخّرة

هذا الكتاب ....

هذا الكتاب ... يرشدك إلى الطريق .

....

فهو يبدأ من أساسيات لغة باسكال ويأخذ بيدك فى سلاسة إلى بناء البرامج المتكاملة الإنشاء .

....

والكتاب يحتوى على عدد كبير من الأمثلة والتمرينات المحلولة فضلاً عن البرامج التطبيقية فى مختلف المجالات الرياضية والتجارية ومعالجة الكلمات

....

لذلك فالكتاب مُوّجه لمختلف مستويات الدارسين كمرجع عربى للغة باسكال وللقراء على اختلاف تخصصاتهم واهتماماتهم .